
Sonderheft Nr. 207
Preis DM 28.–
öS 210.–, sfr 28.–
Elektronik
68172
68173
68174
68452
68153
68590
68802
68488
68487
68451
68461
68851
68454/459
68465
68120/121
68230
68901
68430
68440
68442
68450
68561
68562
68564
68652
68653
68661
68681
68020
68012
68010
68000
68008
68881
Das
68000-
Sonderheft
Hardware – Software – Anwendungen

Arbeitshilfe

Die ELEKTRONIK ist für Sie eine echte Arbeitshilfe, wenn Sie sich mit der Entwicklung und industriellen Anwendung elektronischer Schaltungen und Baugruppen von Geräten und Systemen befassen.



Die ELEKTRONIK liefert Ihnen Informationen zur Konzeption, Projektion, Entwicklung und über die Einführung elektronischer Systeme in die Fertigung.

Die ELEKTRONIK informiert Sie über Bauelemente:
● Speicher aller Art ● Mikroprozessoren ● Logik-Bausteine ● Interface-Bausteine ● Signalprozessoren ● Operationsverstärker ● A/D- und D/A-Umsetzer ● Leistungshalbleiter ● Passive Bauelemente ● Elektromechanische Bauelemente ● Optoelektronische Bauelemente ● Sensoren und Aktuatoren.

Die ELEKTRONIK informiert Sie über Meß- und Prüftechnik:
Die ELEKTRONIK beobachtet die Marktentwicklung, beschreibt technische Grundlagen, gibt Spezialinformationen und macht so den Markt für den Interessenten überschaubar.

Die ELEKTRONIK informiert Sie über Automatisierung:
Besonders intensiv behandelt wird das Gebiet der Klein- und Mittelserienfertigung. Vielfältige Beispiele zeigen, was Industrieroboter in der Automatisierung zu leisten vermögen.

Die ELEKTRONIK informiert Sie über Kommunikationstechnik:
Einige Stichworte:
● Technologien der elektronischen Kommunikation ● Bauelemente für die Telekommunikation ● Übertragungsverfahren ● Meßtechnik ● Normung ● Internationale Zusammenarbeit ● Neue Medien ● Pläne der Fernmeldebehörden.

Die ELEKTRONIK informiert Sie über Schaltungspraxis:
Hier werden erprobte und aktuelle Schaltungen, Entwurfs- und Berechnungshilfen, Meßideen und Anwendungsbeispiele aus dem professionellen und dem industriellen Bereich präsentiert. Beispiele:
● Allgemeine Digitaltechnik ● Interface-Schaltungen ● Signalerzeugung ● Meßtechnik ● Meßwerterfassung und Verarbeitung ● Stromversorgung ● Spezialschaltungen ● µC-Praxis ● Applikationen.

Die ELEKTRONIK informiert Sie über Mikrocomputer:
● Tischcomputer ● Mikroprozessor-CPUs (Bauelemente und Platinen) ● Einchipcomputer ● Mikroprozessor-Peripherie (Bauelemente und Geräte) ● Speicher (Systeme und Bauelemente) ● Software (Systemebene, Anwenderebene, Sprachen) ● Entwicklungssysteme ● Anwendungen, z. B. in der Steuerungstechnik, Medizinelektronik, Konsumelektronik, Meßtechnik ● Praxisnahe Hinweise für Entwickler (µC-Praxis) ● Meß- und Prüftechnik für Mikroprozessorsysteme.

Die ELEKTRONIK informiert Sie über das Produktangebot auf dem internationalen Markt:
● Bauelemente ● Meßgeräte
● Mikrocomputer ● Datentechnik
● Fertigungsmittel ● Software
● Steuer- und Regeltechnik
● Prüftechnik

Die ELEKTRONIK informiert Sie über Aktuelles in der Branche:
Kurzmeldungen aus aller Welt zum Thema ● Technologien ● Verfahren ● Neuheiten ● Unternehmen ● Termine ● Personen

Die ELEKTRONIK informiert Sie mit Sonderpublikationen als Heft im Heft über ● **CAD/CAM** Rechnerunterstützte Entwicklung und Fertigung ● **COM & PRO** Computer und Programme für Anwender von OEM-Produkten ● **TELECOM** Bausteine und Verfahren der Telekommunikation ● **ROBOTER** Flexible Automatisierung in der Industrie

Bitte verwenden Sie zum Kennenlernen unserer ELEKTRONIK die Karte an der hinteren Umschlagseite.



Fachzeitschrift für Entwickler und industrielle Anwender

Vorwort

Mit zunehmenden Möglichkeiten der Halbleitertechnologie lassen sich immer mehr Funktionen auf Silizium integrieren. Dabei entstehen „Superchips“, deren Funktionsumfang so groß ist, daß es selbst erfahrenen Entwicklern nicht leicht fällt, mit diesen Bausteinen umzugehen.

Neben neuen Methoden für die Entwicklungsarbeit mit solchen Chips ist insbesondere eine ausführliche Dokumentation erforderlich, die das Verständnis der komplexen Funktionen fördert.

Die 68000-Familie gehört zu den Bausteinen, die wegen ihres großen Funktionsumfangs an die Dokumentation große Anforderungen stellt.

Antworten auf viele Fragen findet der Entwickler in den Handbüchern, die von den Herstellern herausgegeben werden. Praktische Erfahrungen mit solchen Chips sind die Grundlagen vieler Fachzeitschriften-Beiträge. Gerade hier findet der Praktiker zahlreiche nützliche Hinweise.

Die ELEKTRONIK hat in den letzten Jahren häufig über die 68000-Familie und deren Anwendungen berichtet. Um den Praktiker die zeitraubenden Suche nach Informationen zu ersparen, hat die Redaktion dieses Sonderheft aus bereits veröffentlichten Beiträge zusammengestellt, und diese mit neuen Aufsätzen ergänzt. Damit steht für Entwickler und Anwender von 68000-Systemen eine kompakte, aktuelle Informationsquelle zur Verfügung.

Die Redaktion

Inhalt

Impressum

1985

Franzis-Verlag GmbH, Karlstraße 37–41, 8000 München 2.

Bearbeitet von der Redaktion der Zeitschrift ELEKTRONIK.

Für den Text verantwortlich: Dipl.-Ing. Peter von Bechen.

ISSN 0170-0898

Druck: Franzis-Druck GmbH, München.

Printed in Germany. Imprimé en Allemagne.

ZV-Artikel-Nr. 207021 · F/ZV/985/0050/10'

Thomas W. Starnes

MC68000: Philosophie und praktische Realisierung einer 16/32-Bit-Mikroprozessorfamilie

Als es sich Mitte der 70er Jahre abzeichnete, daß die erfolgreiche Prozessorfamilie MC6800 zukünftig nicht mehr alle Bedürfnisse abdecken konnte, entschloß man sich bei Motorola, unter der Bezeichnung „MACSS“ (Motorola's Advanced Computer System on Silicon) die MC68000-Familie zu entwickeln. Entwicklungsziel war eine leistungsfähige Prozessorfamilie, die auch zukünftigen Ansprüchen Rechnung tragen konnte. Dieser Beitrag zeigt die Philosophie, die in den Produkten der MC68000-Familie verwirklicht wurde.

1 Wortbreite und Registerarchitektur

Es gibt unterschiedliche Kriterien, die einen Prozessor als 8-, 16- oder 32-Bit-Typ qualifizieren. Je nach Hersteller bezieht sich diese Angabe auf die Breite des Datenbusses, des Adreßbusses, der Datenwörter, der internen Datenpfade, der arithmetisch-logischen Einheit (ALU) oder des OP-Codes. Im allgemeinen nimmt man die Breite des Datenbusses als Maß für den ganzen Prozessor, obwohl es eigentlich am besten wäre, die Breite des Op-Codes heranzuziehen. Unter diesem Aspekt ist der MC68000 sowohl ein 16- als auch ein 32-Bit-Prozessor.

Viele Wünsche ergeben ein Konzept

Der Entwickler eines Mikroprozessors legt in hunderten von Einzelentscheidungen die Architektur seines neuesten Produktes fest. Die Anforderungen eines Benutzers müssen als wichtigste Faktoren berücksichtigt werden. Die Benutzer sind es ja schließlich, die das Endprodukt benötigen, wenn sie nicht zufrieden mit Merkmalen oder der Leistung sind, werden sie sich bald nach einer anderen Alternative umschauen.

In der Praxis ist es leider so, daß aufgrund gewisser technischer Grenzen nicht alle Anforderungen eines Benutzers zu erfüllen sind. Der wichtigste Faktor in diesem Zusammenhang sind die technologischen Grenzen, die bei der Herstellung von Bauelementen aus Silizium zu beachten sind.

Bei der Planung des neuen 16-Bit-Prozessors mußte man zunächst die Grundarchitektur festlegen. Wie sollte diese aussehen? Damals existierte bereits eine große Menge Software für den MC6800. Ein Prozessor, der eine Weiterentwicklung gegenüber einem älteren Prozessor darstellt, aber trotzdem noch alle Programme des Vorgängertyps bearbeiten kann, hat einen großen Vorteil: Existierende Software wird nicht wertlos. Allerdings ist die Architektur früherer 8-Bit-Mikroprozessoren alles andere als komfortabel. Diese waren nämlich ursprünglich zum Ersatz einfacher logischer Schaltungen gedacht, so daß deren Befehlssatz mehr an den Anforderungen der Hardware orientiert war. Wenn man

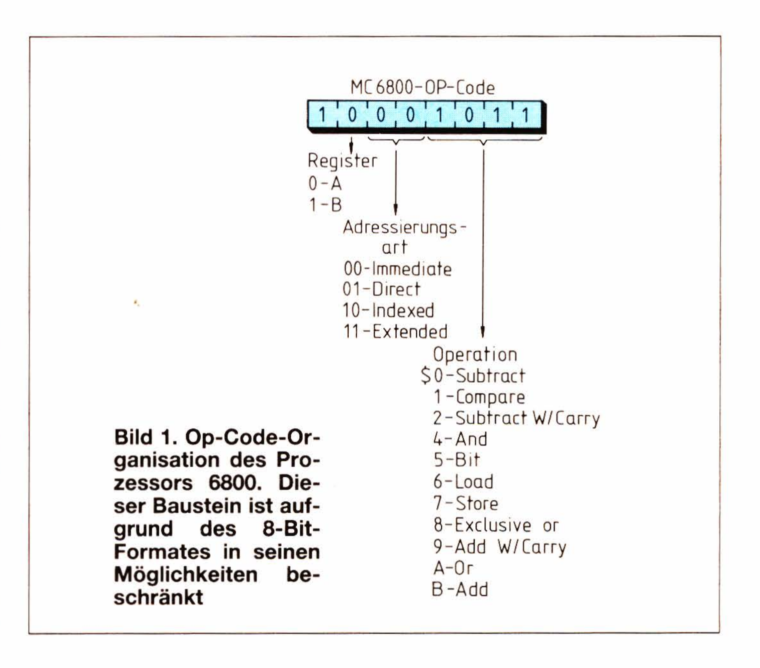


Bild 1. Op-Code-Organisation des Prozessors 6800. Dieser Baustein ist aufgrund des 8-Bit-Formates in seinen Möglichkeiten beschränkt

versucht, einen Mikroprozessor zu entwerfen, der kompatibel mit alten 8-Bit-Typen ist, ist man vom Konzept her sehr stark eingeengt.

Daher entschied man sich bei Motorola, daß das neue Produkt möglichst schnell und flexibel sein sollte. Insbesondere dem Programmierer sollte mit diesem Prozessor die Arbeit erleichtert werden, indem ihm neue Funktionen zur Verfügung stehen. Damit war klar, daß man eine völlig neue Mikroprozessorgeneration zu schaffen hatte, bei der im Vergleich zu den existierenden Produkten ganz andere Konzepte verwirklicht werden mußten.

Der einzige Bereich, der aus der 8-Bit-Zeit noch zu gebrauchen war, waren die bereits vorhandenen Peripheriebausteine. Deshalb entschied man sich, die 68000-Familie so auszulegen, daß die Peripherieeinheiten der MC6800-Familie direkt anschließbar sind. Dies ist auch durchaus sinnvoll, weil viele Ein-/Ausgangs-Operationen eine Wortbreite von 8 Bit erfordern. Außerdem ergab sich der Vorteil, daß zu dem Zeitpunkt, als der 16-Bit-Typ auf den Markt kam, bereits viele Peripheriebausteine zur Verfügung standen.

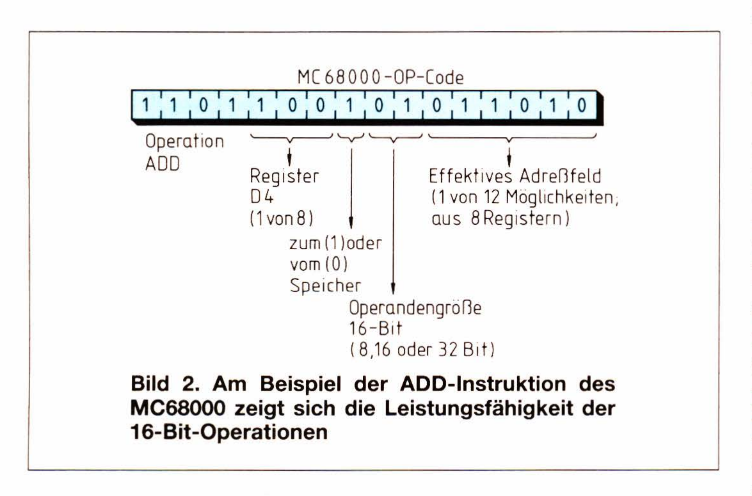


Bild 2. Am Beispiel der ADD-Instruktion des MC68000 zeigt sich die Leistungsfähigkeit der 16-Bit-Operationen

Mehr Möglichkeiten

Ein sorgfältig konzipierter 16-Bit-Mikroprozessor hat gegenüber leistungsfähigen 8-Bit-Typen viele Vorteile, insbesondere in bezug auf die Programmierung (*Bilder 1* und *2*). Weil bei einer Op-Code-Breite von nur 8 Bits lediglich 256 verschiedene Befehle zur Verfügung stehen können, ist das Spektrum der Möglichkeiten begrenzt. Obwohl die Zahl auf den ersten Blick nicht klein ist, müssen folgende Punkte berücksichtigt werden. Wenn ein Mikroprozessor über zwei Register verfügt, aus denen Daten verschoben werden müssen, ist ein Bit für die Decodierung erforderlich. Bei vier verschiedenen Adressierarten für den Zugriff auf den Speicher sind zwei weitere Bits für die Decodierung erforderlich. Damit bleiben für den Mikroprozessor lediglich fünf Bit übrig, mit deren Hilfe sich die Operation festlegen läßt. Das bedeutet, daß lediglich 32 verschiedene Operationen möglich sind. Dies sind eigentlich auch nicht wenig für viele Anwendungen, allerdings stehen nur zwei Datenregister- und vier Speicher-Adressierungsarten zur Verfügung. Ernsthafte Programmierversuche lassen sich damit nur sehr schwer ausführen. Register sind dazu da, daß man Daten sehr schnell manipulieren und Inhalte sehr schnell transferieren kann. Ein leistungsfähiger Mikroprozessor muß daher viele Register besitzen, auf die man mit unterschiedlichen Operationen zugreifen kann.

Darüber hinaus ist der Speicher um so effizienter, je mehr Adressierarten es gibt. Offensichtlich können 8 Bit des Op-Codes dem Mikroprozessor nicht die Vielseitigkeit und die Anzahl von Operationen geben, wie es ein guter 16-Bit-Mikroprozessor kann. Bei 64 000 möglichen unterschiedlichen Befehlen mit einem 16-Bit-Op-Code kann man wesentlich komplexere Operationen definieren und ausführen.

Dies ist der wirkliche Vorteil eines 16-Bit-Mikroprozessors für den Programmierer gegenüber einem 8-Bit-Prozessor. Ein 16-Bit-Typ kann die doppelte Daten-Bus-Breite einer 8-Bit-Ausführung haben. Dieser breitere Bus ermöglicht, daß doppelt so viele Informationen in den Prozessor oder aus ihm innerhalb einer gewissen Zeitspanne zu transferieren sind. Bei einem entsprechenden internen Aufbau bedeutet das die doppelte Menge an Operationen im Vergleich zur 8-Bit-Maschine. 16-Bit-Prozessoren geben dem Programmierer einen größeren Rahmen bei der Codierung und führen vergleichbare Operationen in weniger als der halben Zeit im Vergleich zu einem 8-Bit-Mikroprozessor aus.

Zugriff auf Speicher

Benutzer von 8-Bit-Mikroprozessoren konnten sich zunächst nicht vorstellen, mit welchem Programm man einen Speicherumfang von 64 KByte füllen kann. Viele Systeme hatten früher nicht mehr als 8 KByte ROM und RAM. Als aber mit der Zeit immer mehr Software zur Verfügung stand, waren 64 KByte der Standard für die Speicherkapazität. 16-Bit-Mikroprozessoren stellen in dieser Beziehung noch mehr Ansprüche.

Als der MC68000 entwickelt wurde, war bereits klar, daß ein Adreßbereich von 64 KByte sehr schnell eine Begrenzung darstellt. Jedes zusätzliche Bit für die Adressierung verdoppelt den Speicherbereich, auf den der Prozessor zugreifen kann.

Es gibt verschiedene Techniken, wie man die Grenzen eines 16-Bit-Adreßbereichs überwinden kann (*Bild 3*). Beispielsweise kann man den 16 Adreßbits zusätzliche Bits hinzufügen. Diese befinden sich üblicherweise in einem Register, das die Bezeichnung „Page-Register" trägt. Man nennt die Zugriffsmethode „Paging", weil man immer auf einer bestimmten „Seite" (englisch: Page) arbeitet. Bei früheren Computern mußte diese Seite per Hand eingestellt werden, die unteren 16 Bit der Adresse sind im Befehl oder in Registern enthalten.

Das Paging hat den Vorteil, daß es im Prozessor sehr einfach zu implementieren ist. Im Vergleich zur 16-Bit-

Adressierung ist keine Schaltungsänderung notwendig, weil die gesamte Erweiterung lediglich im Hinzufügen der Bits besteht. Ein weiterer Vorteil ist, daß der Code weiterhin sehr kurz bleibt, weil lediglich 16 Adreßbits in den Befehlen enthalten sein müssen.

Allerdings gibt es auch Nachteile: Ein Programmierer ist auf den Zugriff von nur einer bestimmten Seite im Speicher beschränkt, die im Page-Register angegeben ist. Um sicherzugehen, daß auch die richtige Seite in Benutzung ist, muß geprüft werden, welchen Inhalt dieses Register hat. Dies bedeutet einen gewissen Aufwand für den Programmierer sowie zusätzlichen Code für die Software. Der zusätzliche Code liegt vom Umfang her in der Größenordnung wie die eingesparten Bits bei der Adressierung.

Eine Möglichkeit, die Begrenzung auf eine Seite zu verhindern, ist, mehrere Page-Register vorzusehen. Weitere Merkmale, die davon betroffen sind und die festlegen, welches Register während eines bestimmten Buszyklus aktiv ist, sind z. B. Instruction-Fetch, Data-Read/Write sowie Stack-Access. Während mit zusätzlichen Registern dem Programmierer der Zugriff auf mehr als eine Seite ermöglicht wird, gibt es allerdings nur lediglich eine verfügbare Seite pro Zugriffsart.

Einige Nachteile des Paging lassen sich mit bestimmten Erweiterungen vermeiden. Die Segmentierung, z. B. arbeitet nach dem gleichen Grundprinzip. Der wichtigste Unterschied bei der Segmentierung ist, daß die Page-Nummer eine Segment-Nummer wird und die Segment-Nummer zur 16-Bit-Grundadresse hinzuaddiert wird. Allerdings ist hier auch erforderlich, daß der Programmierer überprüft, ob das gewünschte Segment geladen ist. Jedes Segment ist auf nur 64 KByte begrenzt.

Für den Programmierer ist die einfachste Art der Adressierung die direkte Adressierung jedes Speicherplatzes. Dabei spielt es keine Rolle, ob die gewünschten Daten erst kürzlich benutzt worden waren, oder ob sie weit entfernt sind. Der Programmierer hat einen Überblick über die Daten, das bedeutet, die Spezifizierung ist sehr einfach.

Wenn man über den Adreßumfang eines Prozessors hinaus auf Speicher zugreifen will, ist irgendeine Methode der Speicherverwaltung notwendig. In komfortablen Systemen werden verschiedene Speicherblöcke dynamisch zugeordnet oder angesteuert. Dies erfolgt auch, um gewisse Bereiche schützen zu können, die als Arbeitsbereich vor dem Zugriff anderer Benutzer sicher sein müssen. Das bedeutet, daß eine separate Speicherverwaltungseinheit (MMU – Memory Management Unit) in Zusammenhang mit dem Betriebssystem aus den vorliegenden Informationen die „richtige" Adresse erzeugt. Diese Technik sieht vielleicht ähnlich wie Paging oder Segmentierung aus, erfolgt aber aus einem ganz anderen Grund und auch nach einem anderen Verfahren. Der Programmierer von Anwendungssoftware sieht nichts von der Speicherverwaltung und schreibt seinen Code so, als stände ihm der gesamte Speicher zur Verfügung.

Um den Speicherbereich für den MC68000 zu erweitern, wählte man zwar nicht die am einfachsten auf dem Chip zu implementierende Methode, sondern diejenige, die am einfachsten zu verwenden ist, nämlich einen linearen Adreßbereich. Dieser ist nicht durch irgendwelche Paging-, Segmentierungs- oder Bank-Verfahren unterteilt. Es handelt sich um eine sehr einfache Adressierungsart, die wenig Aufwand für den Programmierer bedeutet, wobei sehr vorteilhafte Operationen, z. B. Speicherverwaltung, durchaus möglich sind.

Eine lineare Adresse umfaßt z. B. 32 Bit. Der Adreßbereich ist nicht in Blöcke aufgeteilt; er ist kontinuierlich. Ein Zugriff auf eine solche Adresse erfolgt mit Hilfe von 32 Bit im Befehl oder unter Verwendung eines einzigen Adreßregisters. Zur Vereinfachung kann man auch im Falle, daß die oberen 16 Bits der Adresse alle auf 0 oder auf 1 gelegt sind, eine 16-Bit-Adresse verwenden. Auf diese Weise greift der MC68000 auf Speicher- und E/A-Adressen zu.

Wie groß sollte der Adreßbereich für einen 16-Bit-Mikroprozessor sein? „Natürliche" Adreßwortlängen sind 16, 24 oder 32 Bits bzw. 2, 3 oder 4 Byte. Wenn man sich überlegt, welchen Umfang zukünftige Mikroprozessorsysteme annehmen können, wird klar, daß sogar die 16 MByte, die sich mit einer 24-Bit-Adresse erreichen lassen, nicht alle Anforderungen erfüllen.

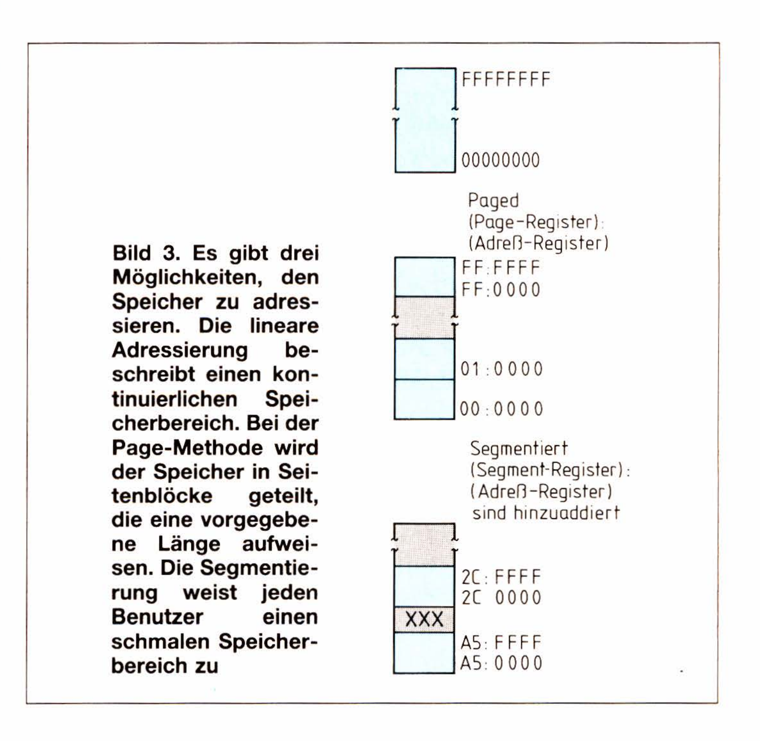


Bild 3. Es gibt drei Möglichkeiten, den Speicher zu adressieren. Die lineare Adressierung beschreibt einen kontinuierlichen Speicherbereich. Bei der Page-Methode wird der Speicher in Seitenblöcke geteilt, die eine vorgegebene Länge aufweisen. Die Segmentierung weist jeden Benutzer einen schmalen Speicherbereich zu

Obwohl 32 Adreßbits, die 4 GByte Speicherkapazität erreichen können, sehr umfangreich zu sein scheinen, wählten die Entwickler des MC68000 einen virtuellen Adreßbereich von 32 Bits.

In bezug auf die praktische Realisierbarkeit sind 32 Adreßanschlüsse jedoch relativ viel. Vor 1980 kannte man kaum integrierte Schaltungen, die mehr als 40 Anschlüsse besaßen, denn üblicherweise verwendete man das Dual-In-Line-Gehäuse. Anfang der 80er Jahre benötigte man nur bei wenigen Systemen Adreßbereiche

von mehr als 16 MByte, so daß man sich entschloß, lediglich 24 Adreßbits nach außen zu führen. Auf diese Weise läßt sich die Zahl der Anschlüsse verringern, so daß sich der MC68000 in einem 64poligen Gehäuse unterbringen läßt. Allerdings sind alle 32 Adreßleitungen im Prozessor vorhanden.

Viele Register erleichtern die Programmierung

Nachdem der Adreßbereich des Speichers festgelegt war, konnte man den Aufbau der Register des neuen Prozessors festlegen.

Die Konfiguration von Registern in einem Prozessor spielt eine wichtige Rolle, weil Daten für die Verarbeitung aus dem Speicher dorthin transferiert werden müssen, was eine gewisse Zeit in Anspruch nimmt. In der Regel müssen Daten mehrmals nacheinander verarbeitet werden, bevor ein Resultat feststeht. Vielfach werden auch verschiedene Teile der Daten miteinander kombiniert. Diese praktischen Erfahrungen führten dazu, daß auf dem Chip Register vorgesehen wurden, die die schnelle Manipulation von häufig benutzten Daten erlauben.

Das Laden und Sichern von Registerinhalten ist normalerweise eine unproduktive Tätigkeit. Die erforderliche Zeit, die nötig ist, um Daten in ein Register auf dem Chip zu bringen, hängt davon ab, was mit den Daten zu geschehen hat. Je mehr Register auf dem Chip zur Verfügung stehen, desto wahrscheinlicher ist es, daß schon in den Registern befindliche Daten für die nächste Operation benötigt werden, so daß kein Rücktransfer zum Speicher erforderlich ist.

Die interne Abarbeitung von Befehlen im Mikroprozessor legt fest, ob sich ein bestimmter Baustein mehr oder weniger zur Programmierung eignet. Befehle können entweder in sogenannten Spezialregistern („Dedicated Registers") oder in allgemeinen Registern (General Registers) abgearbeitet werden. Jede dieser beiden Methoden hat Vor- und Nachteile.

In einem Mikroprozessor mit Spezialregistern enthält der Befehl neben den Daten die Adresse eines speziellen Registers, in dem die Verarbeitung stattfinden soll. Diese Register sind mit der speziellen Instruktion verkettet. Der Befehl „ADD" nimmt beispielsweise nur eine Addition im Register A vor. Wenn der zu addierende Wert nicht im Register A gespeichert ist, muß er zunächst dort hingebracht werden. Davor muß man allerdings den Inhalt von A sichern. Dies alles ist mit sehr großem Aufwand verbunden, der vergleichbar mit dem ist, der bei einer zu geringen Anzahl von Registern anfällt.

Man muß das mit einem Prozessor vergleichen, der allgemein zugängliche Register benutzt. Bei einer solchen Maschine kann der Befehl ADD in jedem der internen Register ausgeführt werden. Dazu muß der Befehl eine Information enthalten, der das Register angibt. Dies geschieht bei der Assemblierung. Wenn der Prozessor vier Register hat, läßt sich die Operation ADD in den Registern A, B, C oder D ausführen, was vom Programmierer festgelegt ist.

Wenn sich der zu addierende Wert im Register C befindet, weist der Programmierer ganz einfach C als Operandenregister zu. Der Inhalt muß nicht transferiert werden, außerdem ist kein Registerinhalt zu sichern. Eine solche Maschine ist daher wesentlich einfacher zu programmieren und arbeitet schneller bei der Ausführung einer bestimmten Operation.

Die einfache Programmierung muß man allerdings mit einem Nachteil erkaufen: Die Auswahl des Registers erfordert einige Bits im Op-Code, so daß dieser länger wird. Außerdem ist es schwieriger, eine Schaltung mit verschiedenen Registern zu implementieren, weil es eine gewisse Zeit erfordert, festzulegen, welches Register benutzt werden soll. Die zusätzliche Zeit und der Mehraufwand zahlen sich allerdings durch die höhere Flexibilität bei der Programmierung wieder aus.

Aus diesem Grund wurde der MC68000 mit universellen Registern konzipiert. Jeder Befehl läßt sich in Zusammenhang mit jedem Register als Ursprungs- oder Bestimmungsort benutzen oder als Pointer in Zusammenhang mit jeder zulässigen Adressierungsart. Dieser hohe Grad an Flexibilität gibt den Programmierern alle Möglichkeiten zur Anordnung von Daten und Pointern. Wenn man die Benutzung von Registern untersucht, zeigt sich, daß sie in der Regel einem von zwei Zwecken dienen: Sie enthalten entweder die Daten für eine bestimmte Manipulation oder eine Adresse, die auf einen Speicherplatz hinzeigt. Die Verwendung eines Registers unterscheidet sich bei diesen beiden Möglichkeiten. Wenn Daten in oder aus dem Register transferiert werden oder eine Manipulation im Register vorgenommen wird, sind alle Arten an Bedingungsinformation von der Operation wichtig. Daher müssen alle „Condition Codes" nach der Operation richtig gesetzt sein. Auf diese Weise lassen sie sich zur Verzweigung oder in Zusammenhang mit anderen Datenoperationen verwenden.

Auf der anderen Seite läßt sich eine Adresse in einem Register ablegen oder von dort herholen, wobei eine Modifizierung, z. B. Inkrementierung oder Dekrementierung möglich ist. Manchmal ist es wichtig, ob ein Übertrag aus der ALU kommt oder ein Resultat negativ ist. In der Praxis zieht ein Programmierer die Manipulation einer Adresse vor, die keinen Einfluß auf die Condition-Codes hat. Manchmal benötigt man mitten in einer komplexen Datenoperation eine neue Adresse oder muß eine Adresse inkrementieren. Dabei ist es unerwünscht, daß im Zuge dieser Aktion die Condition-Codes sich verändern.

Aus diesem Grund gibt es zwei Grund-Registertypen: Datenregister (D0...D7) und Adreßregister (A0...A7). In einem Datenregister betrifft jede Operation die Condition-Codes des Mikroprozessors. Dagegen werden in einer Operation des Adreßregisters die Condition-Codes nicht beeinflußt, die Codes von vorhergehenden Datenoperationen sind gesichert. Auf diese Weise lassen sich Adreß- und Index-Pointer-Veränderungen durchführen, auch wenn sie mitten in komplexen Datenoperationen erforderlich werden.

Wie viele Register sollten vorgesehen sein und welche Größe sollten sie haben? Je mehr Register zur Verfügung stehen, desto besser ist das für den Programmierer. Andererseits steigt der Preis des Chips mit wachsender Registerzahl und der dafür erforderlichen Steuerschaltung. Zwischen diesen beiden Forderung muß ein guter Kompromiß gesucht werden.

Zwei Register sind zu wenig, vier sind sehr gut und eine komplexe Routine, die mehr als acht verschiedene Pointer benötigt, ist kaum vorzustellen. Die Codierung von acht Registeradressen erfordert drei Bits. Dieser Umfang ist durchaus vertretbar, so daß man beim MC68000 acht Adreßregister und dazu acht Datenregister vorsah.

Mit den 16 zur Verfügung stehenden Registern, von denen jeweils die Hälfte für Daten und die andere Hälfte für Adressen vorgesehen ist, wird wohl kaum die Notwendigkeit für Zwischenspeichern eines Registerinhaltes erforderlich werden. Die Manipulation von Speicherpointern in den Adreßregistern beeinflußt außerdem nicht die laufende Berechnung der Daten. Auf diese Weise ist der MC68000 einfach zu programmieren.

Es wurde bereits erwähnt, daß der MC68000 Adressen von einer Breite von 32 Bit verarbeiten kann. Jeder, der jemals 8-Bit-Mikroprozessoren mit 8-Bit-Akkumulatoren und 16-Bit-Indexregistern programmiert hat, kennt die Schwierigkeiten mit den beiden unterschiedlichen Wortbreiten. Nachdem man erst einmal herausgefunden hat, wie man den 16-Bit-Wert in die beiden 8-Bit-Akkumulatoren bringt, steht man vor Problemen, die bei dem Versuch entstehen, arithmetische Überträge von der unteren zur oberen Hälfte des Wertes zu bringen.

Diese Erfahrung führte beim Entwurf des MC68000 dazu, daß die Adreßregister und die Datenworte gleiche Breite aufweisen. Um einen linearen virtuellen 32-Bit-Adreßbereich verwalten zu können, muß der MC68000 über 32-Bit-Adreßregister verfügen. Man erwartet von einem 16-Bit-Mikroprozessor, daß er 8- und 16-Bit-Daten verarbeitet. Warum sollte er dann auch 32-Bit-Daten verarbeiten? Ganz offensichtlich müssen Adressen dieser Größe behandelt werden. Viele Entwickler haben die Erfahrung gemacht, daß die Möglichkeit, mit einem 8-Bit-Mikroprozessor auch 16-Bit-Daten zu verarbeiten, sehr hilfreich ist, insbesondere bei anspruchsvollen Anwendungen. Genauso wie 8-Bit-Prozessoren für die Verarbeitung von 16-Bit-Operanden erweitert wurden, besteht die Notwendigkeit von 32-Bit-Operationen bei 16-Bit-Computern.

Wenn erst einmal wenige 32-Bit-Operationen in einem Mikroprozessor zur Verfügung stehen müssen, wünscht ein Anwender diese Wortbreite auch für alle anderen Operationen. Wenn eine Multiplikation ein 32-Bit-Ergebnis erzeugt, muß man, um mit diesem Ergebnis etwas anfangen zu können, andere 32-Bit-Operationen vorsehen. Aus diesem Grund entschied man sich, daß die Datenregister eine Breite von 32 Bit aufweisen sollen und daß Operationen mit allen 32 Bits in einem einzigen Befehl zu erledigen sind.

Drei Arithmetikeinheiten

Nach sorgfältiger Analyse der Anforderungen an einen modernen Prozessor kam man auf eine Architektur mit drei separaten Arithmetikeinheiten, die parallel arbeiten können. Der Prozessor MC68000 verfügt zunächst über eine 16-Bit-ALU, die die gesamten Datenberechnungen durchführt. Bei den beiden anderen ALUs handelt es sich ebenfalls um 16-Bit-Ausführungen. Sie werden für die verschiedenen Adreßberechnungen in Zusammenhang mit Operanden benötigt. Dies ist sinnvoll, weil alle Adressen eine Breite von 32 Bit haben. Eine effektive Adresse (EA) ist das Resultat einer Berechnung, die sich aus der gewählten Adressierungsart des Prozessors ergibt. Weil bei der Bestimmung der EA ein merklicher Anteil der Gesamtzeit für den Befehl benötigt wird, ist es wichtig, diesen Vorgang möglichst schnell auszuführen.

Aus diesem Grund lassen sich auf diese Weise gleichzeitig eine 32-Bit-Adresse und ein 16-Bit-Datum berechnen. Dies beschleunigt die Verarbeitung einer Instruktion merklich.

Der Prozessor MC68000 verarbeitet auch 32-Bit-Daten. Dies erfolgt normalerweise mit Hilfe von zwei Durchläufen mit 16-Bit-Daten, und zwar eine obere Worthälfte und eine untere Worthälfte. Dies macht sich in der Verarbeitungsgeschwindigkeit bemerkbar.

„Prefetch-Queue"

Eine andere Möglichkeit, den Prozessor MC68000 schneller zu machen, ist die sogenannte „Prefetch-Queue". Es handelt sich um eine Warteschlange für das vorgezogene Holen des Befehls, die wesentlich intelligenter als bisherige Techniken dieser Art ist. Die Steuerung erfolgt in Abhängigkeit des Inhaltes im Befehlsstrom.

Die Prefetch-Queue ist eine sehr effektive Möglichkeit, die Leistung des Mikroprozessors zu erhöhen. Mit ihr versucht man, möglichst viele Informationen über den Befehl verfügbar zu haben, bevor dieser ausgeführt wird. Der Mikroprozessor benutzt dazu einen sonst leeren Datenbus, um die Instruktion vorbereitend zu holen.

Der Bereich des Speichers, aus dem die Instruktionen geholt werden, der Programmbereich, enthält Op-Codes und Adressierungsinformationen.

In der Prefetch-Queue ist genug Information enthalten, um eine Instruktion auszuführen, die nächste zu decodieren und die darauf folgende Instruktion aus dem Speicher zu holen – und das alles gleichzeitig.

In bezug auf die Befehlssequenzen ist die Queue variabel. Damit soll verhindert werden, daß unnötig Zeit vergeht. Wenn beispielsweise ein Befehl zum bedingten Sprung erkannt wird, ist der Prozessor bereits auf die Verzweigung vorbereitet, wenn die Entscheidung fällt. Die Warteschlange versucht, sowohl den Op-Code nach dem Verzweigungsbefehl sowie den Op-Code am berechneten Verzweigungsziel zu erhalten. Wenn dann die Condition-Codes verglichen sind und eine Entschei-

dung gefallen ist, ob ein Sprung ausgeführt werden soll, beginnt der Prozessor direkt mit der Decodierung der betreffenden Instruktion. Der überflüssige Op-Code wird jeweils ignoriert.

Man kann die Prefetch-Queue auch in vielen anderen Sonderfällen benutzen. Ein Beispiel ist die Beschleunigung der wiederholten „Move-Multiple-Register"-Instruktion, bei der sich aufeinander folgende Daten-Transfers schneller ausführen lassen. Die Prefetch-Queue führt dazu, daß viele Instruktionen genau in der Zeit ausgeführt werden, die benötigt wird, um den Op-Code aufzunehmen (das ist in Wirklichkeit die Zeit, um den nächsten Op-Code vorbereitend zu holen).

Mikrocodierung

Eine weitere wichtige Entscheidung bei der Entwicklung des MC68000 war, ob man eine sogenannte Random-Logik oder eine mikrocodierte Struktur benutzt. Beide Techniken haben ihre Vor- und Nachteile. Frühere Mikroprozessoren waren zum größten Teil mit Random-Logik aufgebaut. Die modernen Techniken der Hochintegration (VLSI) sowie die zunehmende Komplexität der Chips lassen aber die Mikrocodierung günstiger erscheinen.

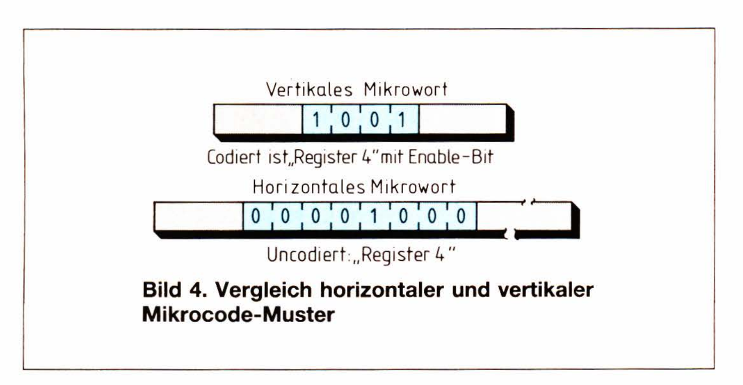


Bild 4. Vergleich horizontaler und vertikaler Mikrocode-Muster

„Random-Logik" bedeutet, daß ein Mikroprozessor oder ein anderer Logikbaustein aus diskreten Funktionselementen, z. B. Gattern, Puffern und Transistoren, aufgebaut ist. Hierbei sind nur die Elemente erforderlich, die auch wirklich benötigt werden. Es gibt keine unbenutzten Gatter oder doppelte Schaltungsteile. Ein solches Konzept führt zu sehr dicht gepackten Schaltungsaufbauten, die auch sehr schnell arbeiten können.

Die Schwierigkeit bei einem solchen Konzept ist bei wachsender Komplexität die Planung und das Layout der Funktionselemente und Signalleitungen. Das bedeutet, daß ungeheuer viel Zeit für die Entwicklung solcher Schaltungen erforderlich wird.

Ein anderes Problem bei dieser Art Schaltungstechnik ist die Durchführung der Erprobungsphase und von Tests. Bevor eine Schaltung in Silizium realisiert wird, muß sie eine Erprobungsphase durchlaufen, die als Simulation auf einem Computer durchgeführt wird. Bei einer Random-Logik-Schaltung muß der gesamte Baustein simuliert werden, um sicherzustellen, daß alle Kombinationen der Signale definiert arbeiten.

In ähnlicher Weise muß, wenn die Schaltung als Siliziumbaustein vorliegt, ein Test vorgenommen werden, der bei einem Random-Logik-Chip sehr schwierig ist. Weil man auf viele Bereiche des Chips nur über lange sequentielle Befehlsreihen zugreifen kann (die Anzahl der Gehäusestifte ist ja begrenzt), kann es vorkommen, daß gewisse Fehler nicht direkt erkannt werden.

Hier bietet sich die Möglichkeit der Mikroprogrammierung zur Lösung des Problems an. Ein Mikroprozessor mit Mikroprogrammierung besitzt einen Sequenzer, der den Datenfluß durch die verschiedenen Funktionsgruppen (ALU, Register, Condition-Flags, Schieberegister, Busse usw.) entsprechend der mikroprogrammierten Befehle leitet. Jeder Befehl hat seine eigene Mikroroutine, d. h. Sequenz von Mikrowörtern, die die dazugehörenden Daten zur entsprechenden Funktionsgruppen in der richtigen Reihenfolge leitet. Auch bedingte Sprünge und Verzweigungen sind möglich.

Mikrocodierung einer komplexen Schaltung erleichtert den Entwurf, weil sie dazu führt, daß die Schaltung modular strukturiert ist. Benötigt werden ein Controller, ein Mikroprogramm-Block sowie die Elemente, durch die die Daten fließen. Jedes dieser Elemente läßt sich mit individuellen Ein- und Ausgängen ausstatten und daher unabhängig aufbauen und testen. Daher hat die Mikrocodierung zu einer wesentlichen Vereinfachung des Entwurfprozesses geführt.

Ein anderer Vorteil der Mikrocodierung ist die große Flexibilität beim Betrieb der Schaltung. Die Mikrowörter erlauben mehr Kombinationen der einzelnen Ein- und Ausgänge als eine herkömmliche Random-Logik. Insbesondere für den Entwickler von ICs ist die Mikrocodierung interessant, denn Änderungen bestehender Chipentwürfe sind wesentlich einfacher durchzuführen.

Änderungen in letzter Minute möglich

Bei der Herstellung der ICs lassen sich die Mikrobefehle praktisch bis zu dem Zeitpunkt ändern, bevor die Masken für die Herstellung der Halbleiterstruktur fertiggestellt sind. Die Modifizierung einer kleinen Einzelheit im Betriebsablauf bedeutet lediglich das Ändern einiger weniger Bits im Mikrocode-ROM.

Der Nachteil mikrocodierter Schaltungen liegt im modularen Aufbau begründet, für den wesentlich mehr Transistoren und damit eine größere Fläche der integrierten Schaltung erforderlich ist. Im Vergleich zur herkömmlichen Logik rechnet man mit etwa 20 % mehr Chipfläche. Allerdings wird dieser Nachteil von den eben beschriebenen Vorteilen bei weitem aufgewogen, insbesondere seitdem die Probleme moderner hochintegrierter Schaltungen beherrscht werden.

Es gibt zwei Arten der Mikroprogrammierung: horizontale und vertikale (*Bild* 4). Bei der horizontalen Mikrocodierung handelt es sich um die direktere Art. Sie ist nicht codiert, so daß z. B. ein Bit in jedem

Die 68000 Familie

63484 Graphic Controller
- 38 high level Befehle
- On-Chip 32 Byte Pattern RAM
- alle CRT Timing Signale programmierbar

HD 63484

63463 Hard Disk Controller
- ST506 und SMD Interface
- 2 × 256 Bytes On-Chip Buffer
- CMOS

HD 63463

68450 DMA Controller
- 4 unabhängige Kanäle
- 2 Interrupts pro Kanal
- 4 MByte/Kanal

HD 68450

50256 50256 50256

Speicher
- 2 µ CMOS/NMOS
- hohe Packungsdichte (LCC, PLCC, SOP)
- hohe Integration

HD 68000

68000 16/32 Bit Mikroprozessor

- jetzt auch in CMOS
- Shrink Package
- Plastik
- mächtiger Befehlssatz



A World Leader in Technology

Hitachi Electronic Components Europe GmbH
Hans-Pinsel-Straße 10A · 8013 Haar b. München
Tel. 089/46 14-0 · Tx. 522 593 · Fax 089/46 31 51

Vertriebsbüros
Neuss: Breslauer Straße 6 · 4040 Neuss 1 · Tel.: 0 21 01/15 00 27-29
Stuttgart: Fabrikstraße 17 · 7024 Filderstadt 4 · Tel.: 07 11/77 20 11
München: Hans-Pinsel-Str. 10A · 8013 Haar · Tel.: 089/46 14-0

Distributoren:
PLZ 2 + 3
Henskes GmbH, Tel. 05 11/45 60 82
PLZ 4 + 5
H. M. Müller, Tel. 02 02/42 60 16
PLZ 6 + 7 + 8
MSC, Tel. 0 72 49/70 75
PLZ 7
Ditronic GmbH, Tel. 07 11/72 00 10
PLZ 8
Data Modul GmbH, Tel. 089/56 01 70

Schweiz
Fenner Elektronik AG, Tel. 061/98 22 02
Österreich
Ing. Ernst Steiner, Tel. 02 22/8 27 47 40

Mikrowort ein Register freigibt. Bei 16 Registern sind daher 16 Bit Mikrocode erforderlich. Horizontale Mikrowörter werden relativ lang, sie erfordern daher größere Chips und höhere Kosten.

Kompakter, allerdings auch etwas langsamer ist die vertikale Mikrocodierung. Hier werden die Steuerfunktionen codiert, so daß lediglich 4 Bit für die Auswahl von einem aus 16 Registern erforderlich sind. Die vertikale Mikroprogrammierung benötigt wenigstens eine Gatterebene zur Decodierung der Signale. Dieses wiederum bedeutet, daß sich die Verarbeitung verzögert.

Beim MC68000 entschloß man sich zur Mikrocodierung. Im Nachhinein war dies eine richtige Entscheidung. Die ersten Chips arbeiteten seinerzeit bereits zufriedenstellend, so daß sich die wichtigsten Schaltungsteile testen ließen. Die darauf folgenden Modifizierungen waren in der Regel lediglich Korrekturen des Mikrocodes.

Um sowohl die Vorteile der horizontalen als auch der vertikalen Mikrocodierung nutzen zu können, entschloß man sich, eine Kombination aus beiden zu nehmen. Man entwickelte einen sogenannten „Mikrocode" und einen „Nanocode". Beim Mikrocode handelt es sich um eine Serie von Pointern, die auf bestimmte Mikro-Subroutinen des Nanocodes zeigen. Letzterer übernimmt die eigentliche Steuerung und Auswahl der Register sowie Funktionen und sorgt für die Weiterleitung der Resultate. Eine solche Kombination ist sehr effektiv, weil der größte Teil des Codes gewisse Routinen gemeinsam benutzen kann, wobei die Individualität verschiedener Instruktionen nicht verlorengeht.

Die Decodierung eines Op-Codes erzeugt die Startadresse im Mikrocode für den Operationstyp und die Adressierungsart. Der Abschluß einer Instruktion gibt die Möglichkeit frei, Interrupts entgegenzunehmen oder ermöglicht Zugriff auf die Prefetch-Queue für den nächsten Op-Code. Die Prefetch-Queue benötigt den Bus in 85...95 % der Zeit, das heißt der Bus ist lediglich 5...15 % der Zeit unbenutzt.

2 Daten-Verschiebung, arithmetische und logische Befehle

Um die Befehle des MC68000 bis ins Detail verstehen zu können, empfiehlt es sich, das Benutzerhandbuch genau zu studieren. In diesem Beitrag soll ein genereller Überblick über die Befehle gegeben werden. Anhand von speziellen Beispielen erkennt man die Leistung des Befehlssatzes.

Befehlsformat und Adressierungsarten

Bevor auf die Befehlsarten eingegangen werden soll, zunächst ein Blick auf Befehle in Assemblersprache. *Tabelle 1* zeigt ein allgemein übliches Befehlsformat und die Auswahlmöglichkeiten, die zur Verfügung stehen. Zunächst wählt man einen der verschiedenen Mikroprozessorbefehle, z. B. eine Addition (ADD), Vergleich (CMP), arithmetischer Schiebevorgang links (ASL) oder Daten-Verschiebung (MOV). Wenn es sich um einen Datenverarbeitungsbefehl handelt, besteht beim MC68000 die Möglichkeit, diese mit 8, 16 oder 32 Bit auszuführen. Die Auswahl erfolgt durch folgenden mnemonischen Code „B", „W", „L" (Byte, Wort, Lang-Wort). Wenn keine Größe spezifiziert ist, geht der Assembler von einer 16-Bit-Operation aus.

Bei einer Datenoperation gibt es eine oder mehrere Möglichkeiten, das heißt Adressierarten für den einen oder die zwei Operanden, die für die Instruktion erforderlich sind. In der Praxis kann man eine aus 14 verschiedenen Betriebsarten auswählen, dabei steht in den meisten dieser Fälle eine von acht Adreßregistern zur Verfügung. Bei vielen Operationen muß auch noch eine zweite Adressierart gewählt werden, was üblicherweise die Auswahl eines der acht Datenregister beinhaltet. Für die Daten-Verschiebungsbefehle ist jede Adressierungsart zulässig.

Alle Befehle des MC68000 sind mit den 16 Bits des Op-Codes definiert. Abhängig vom Befehl oder der Adressierart können zusätzliche Erweiterungswörter dem Op-Code folgen. Diese bestehen aus zusätzlicher Adressierungsinformation. Die Gesamtlänge eines Befehls kann bis zu 10 Byte betragen. Weil ein Befehl immer ein Mehrfaches von 16 Bit lang ist, beginnt jede Instruktion bei einem gradzahligen Byte.

Am häufigsten bei jeder Prozessoranwendung ist die Verschiebung von Daten. Andere Mikroprozessoren verschieben ihre Daten mit den Befehlen Load, Sore, Push, Pull, Pop sowie Ein-/Ausgabe-Instruktionen. Das wesentliche dabei ist das Verschieben der Daten von einem zu einem anderen Ort. Es spricht also nichts dagegen, diesen Vorgang mit „MOVE" zu bezeichnen. Dies trägt auch zur Erleichterung des Verständnisses bei.

Die verschiedenen MOVE-Befehle

Die MOVE-Instruktion kann 8-, 16- oder 32-Bit-Daten von praktisch jedem Ort zum beliebigen anderen Ort transferieren. Ein breites Spektrum an Adressierarten

Tabelle 1. Allgemeines Format für 68000-Befehle

Befehlsformat:
Mnemo-Code . Größe   Quelle , Bestimmungsort

Beispiel:

```
ADD.L     D1,D2
MOVE.B    #15,-1(A0)
ADD       D1,D2 (Vorausgesetzt Größe: W)
BGE       L00P1 (nur ein Argument)
RTS             (keine Argumente)
```

Erklärung:

Mnemo-Code:	Abkürzung für Befehle
Größe:	.B – Byte (8 Bit)
	.W – Wort (16 Bit)
	.L – Langwort (32 Bit)

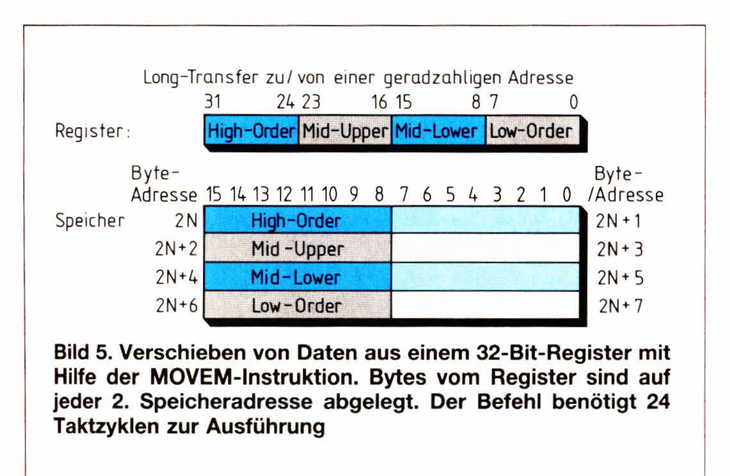


Bild 5. Verschieben von Daten aus einem 32-Bit-Register mit Hilfe der MOVEM-Instruktion. Bytes vom Register sind auf jeder 2. Speicheradresse abgelegt. Der Befehl benötigt 24 Taktzyklen zur Ausführung

sowie Register, die als Ursprung und Bestimmungsort in Frage kommen, decken alle Möglichkeiten ab. *Tabelle 2* zeigt die verschiedenen Kombinationen für den Typ MC68000 im Vergleich zur 8086-Familie.

Was kann man mit diesen Möglichkeiten tun? Zunächst lassen sich Daten zwischen Registern kopieren, aber auch Daten von einem Register in einen Speicher bringen, in dem man Speicher-Adressierarten nutzt. Viele Mikroprozessoren ermöglichen lediglich den Transfer von Daten vom „Top" des Stacks oder von einem Register. Was kann man tun, wenn man die Daten an einer anderen Stelle im Speicher benötigt? Dazu ist ein zweiter Befehl notwendig. Beim MC68000 gibt es in dieser Beziehung keine Probleme: Man kann die Daten des „Top-of-Stack" zu oder von jedem Register, einem anderen Stack, einer Warteschlange, einem Speicherplatz oder sogar E/A-Anschluß transferieren, und das alles in einer einzigen Bewegung. Möglich wird dies dadurch, daß man eines der acht Adreßregister als Stack-Pointer verwenden kann. Dies erlaubt den Aufbau von nicht weniger als acht unterschiedlichen Stacks, ohne daß die Register in den Speicher umgeladen werden müssen.

Datenverschiebungen zwischen zwei Speicherplätzen sind ebenfalls möglich. Es gibt zehn verschiedene Speicher-Adressierarten zur Auswahl des Quellen-Operanden und sieben für den Bestimmungsort. Jede Adressierart kann dazu jedes beliebige der acht Adreßregister verwenden, so daß sich die Vielseitigkeit dieser MOVE-Befehle noch erweitert.

Wenn man einmal zusammenrechnet, wie viele Möglichkeiten es gibt, Daten in den MC68000 zu bekommen, kommt man auf 34 888 unterschiedlichen Versionen, die jeweils für 8-, 16- oder 32-Bit-Daten zu nutzen sind.

Zu den übrigen Daten-Verschiebungsbefehlen gehört z. B. „SWAP", mit dessen Hilfe sich der Inhalt von zwei Daten- und/oder Adreßregistern austauschen läßt. Mit „MOVE SR" (Status-Register) läßt sich der Inhalt des Status-Registers lesen oder modifizieren.

Der MC68000 ist, wie bereits erwähnt, direkt zum Anschluß an die 8-Bit-Peripherieeinheiten aus der MC68000-Familie ausgelegt. Um dieses zu erleichtern, ist im Befehlssatz des MC68000 die Instruktion „MOVEP" (Move Peripheral) vorgesehen.

Bild 5 zeigt die Funktionsweise. In der Regel muß man ein Register setzen, um die Peripherieeinheit für den Betrieb vorzubereiten. Dazu ist ein 8-Bit-Peripheriebaustein entweder an der oberen oder unteren Hälfte des 16 Bit breiten Datenbusses anzuschließen. Dies bedeutet, daß die Register, die mit dem Peripheriebaustein verbunden sind, innerhalb des Speicheradreßbereiches des MC68000 als aufeinanderfolgende geradzahlige oder ungeradzahlige Adressen erscheinen. Der Befehl MOVEP schiebt entweder 16 oder 32 Bit eines vorgegebenen Datenregisters aus dem Speicher in 8-Bit-Blöcken, beginnend an einer bestimmten Adresse. Die Adressen für die darauffolgenden Bytes werden um zwei erhöht. Damit ist es möglich, daß die 2 oder 4 Byte zu den richtigen Portadressen gelangen und daß nicht

Tabelle 2. Adressierungsarten für die Prozessoren MC68000 und 8086

	Destination #Optionen	Dn 8	An 8	(An) 8	(An)+ 8	−(An) 8	d16(An) 8	d8(An,Xn) 128	Abs.W N/A	Abs.L N/A
Source	#Optionen									
Dn	8	MI	MI	MI	M	M	MI	MI	MI	M
An	8	MI	MI	MI	M	M	MI	MI	MI	M
(An)	8	MI	MI	MI	M	M	M	M	M	M
(An)+	8	M	M	M	M	M	M	M	M	M
−(An)	8	M	M	M	M	M	M	M	M	M
d16(An)	8	MI	MI	M	M	M	M	M	M	M
d8(An,Xn)	128	MI	MI	M	M	M	M	M	M	M
Abs.W	N/A	MI	MI	M	M	M	M	M	M	M
Abs.L	N/A	M	M	M	M	M	M	M	M	M
d16(PC)	1	M	M	M	M	M	M	M	M	M
d8(PC,Xn)	16	M	M	M	M	M	M	M	M	M
Immediate	1	M	M	M	M	M	M	M	M	M

Erläuterungen:
„M" bedeutet, daß diese Kombination auf dem 68000 verfügbar ist
„I" bedeutet, daß eine vergleichbare Kombination bei der 8086-Familie verfügbar ist
„#Optionen" gibt die Zahl der verschiedenen Adressierungsarten an, die vom 68000 benutzt werden können

weniger als vier Register mit jeweils 8 Bit mit nur einem Befehl geladen werden. Der Befehl MOVEP ist bidirektional, so daß sich Register sowohl laden als auch lesen lassen.

Zwei Spezialtypen des MOVE-Befehls sind MOVEQ (Move Quick) und MOVEM (Move Multiple Register). Sehr häufig wird ein Registerinhalt lediglich gezählt oder als Konstante benutzt, mit Werten, die typischerweise relativ klein sind. Der Befehl MOVEQ macht es sehr einfach, ein Register auf solche Werte einzustellen. MOVEQ nimmt jeden 8-Bit-Immediate-Wert mit Vorzeichen zwischen −128 und +127, erweitert das Vorzeichenbit, so daß die Zahl als 32-Bit-Zahl korrekt interpretiert wird und lädt diese anschließend in eines der Datenregister. Der Op-Code für MOVEQ umfaßt den 8-Bit-Immediate-Wert, was bedeutet, daß der Mikroprozessor die Operation sehr schnell ausführen kann. Weil der kleine Immediate-Wert Teil des MOVEQ-Op-Codes selbst ist, ist dieser Befehl als eigene Adressierart klassifiziert. Sie trägt die Bezeichnung „Quick Immediate".

Üblicherweise muß man bei der Programmierung in Maschinensprache den Inhalt verschiedener Register auf dem Chip sichern, die Register für andere Zwecke benutzen und anschließend den vorherigen Inhalt zurückladen. Dies ist am Anfang oder Ende eines Unterprogramms notwendig, wenn ein Interrupt-Handler abgearbeitet wird, die Tasks geändert werden oder beim Aufruf des Betriebssystems. Für diesen Zweck verfügt der MC68000 über eine sehr hilfreiche Instruktion, die diesen Vorgang in Form einer schnellen, effizienten Operation ausführt. Der Befehl „MOVEM" übernimmt jede Kombination der Inhalte von 16 Daten- und Adreß-Registern und transferiert sie von oder zum Speicher. Dabei ist sichergestellt, daß jedes Register den richtigen Inhalt bekommt. Eine Option des Befehls MOVEM ist, daß entweder die unteren 16 Bits oder alle 32 Bits eines Registers sich transferieren lassen. Ein Beispiel für diesen Befehl ist:

MOVEM.L DO/D4–D7/A4/A5,40(A6)

Dieser Befehl sichert die Registerinhalte, wie das in *Bild* 6 zu erkennen ist (es werden die Register D0, D4...D7, A4 und A5 in den Speicherbereich gebracht, der bei einer Adresse beginnt, die sich aus dem Registerinhalt A6 + dem Wert 28 – hexadezimal – ergibt). Die Liste der Register, die zu transferieren ist, wird kompakt in einen 16-Bit-Wert codiert, der dem MOVEM-Op-Code-Wort folgt. Ein „ON"-Bit zeigt, daß das entsprechende Register zu transferieren ist. Die Instruktion MOVEM ist nicht nur sowohl kompakt als auch sehr nützlich, sie ist darüber hinaus für die zu übertragende Anzahl von Informationsbytes sehr schnell.

Orthogonalität

Arithmetische Operationen sind die wichtigsten Befehle in einem Mikroprozessor, denn mit ihnen muß der größte Teil der Arbeit erledigt werden. Die arithmetischen und logischen Instruktionen ermöglichen es einem Programmierer, den Code so zu schreiben, wie er gebraucht wird, ohne daß er die Daten neu anordnen, zusätzliche Daten erfassen oder Dinge in einer unnatürlichen Weise abarbeiten muß. Wie bei vielen anderen Merkmalen des MC68000 ist der Aufbau der arithmetischen und logischen Befehle orthogonal, mehr als bei jedem Mikroprozessor, der vorher auf den Markt kam.

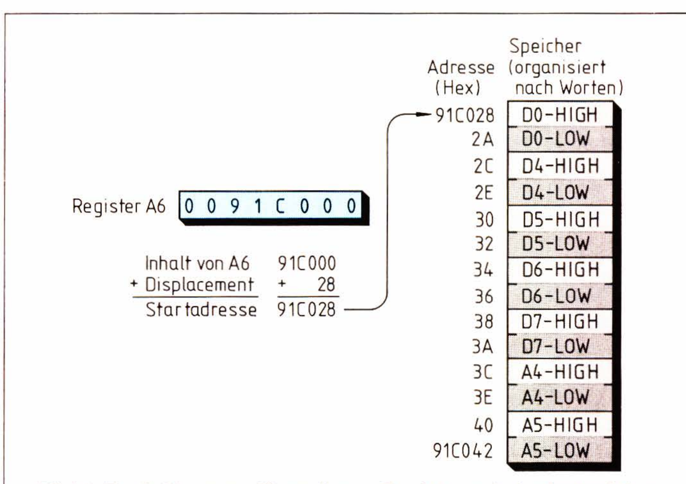


Bild 6. Push-Vorgang für mehrere Register mit der Instruktion MOVEM. Hierbei ergibt sich eine sehr schnelle Methode, bestimmte Registerinhalte im Speicher abzulegen und sie später korrekt wiederzurückzuerhalten, in diesem Fall handelt es sich um die Register D0, D4 bis D7, A4 und A5, deren Inhalt in den Speicherbereich gebracht wird, der mit 91C028 beginnt. Der Befehl lautet: MOVEM. L D0/D4–D7/A4/A5,40 (A6). Für die Ausführung sind 58 Taktzyklen erforderlich

Orthogonalität läßt sich als die Möglichkeit definieren, daß Operationen jede Ressource auf jede Weise nutzen können.

Die arithmetischen und logischen Befehle sind sehr ähnlich in ihrer Funktion, z. B. wie die Condition-Codes verändert werden und welche Auswahl von Adressierungsarten, Register und Operanden-Variablen möglich sind. Der Vorteil dabei ist, daß ein Programmierer bei der Codierung lediglich eine Grundmenge an Regeln beachten muß. Andere Mikroprozessoren schreiben bestimmte Regeln für ähnliche Befehle vor, wodurch die Produktivität eines Programmierers negativ beeinflußt wird.

Alle arithmetischen Doppel-Operanden-Instruktionen sind echte 1½-Adreß-Operationen (d. h. ein Operand kann mit Hilfe einer Speicheradresse spezifiziert werden, aber der andere muß sich in einem internen Register befinden. Das Resultat überschreibt einen Operanden). Auf diese Weise kann man jeden Registerinhalt zu jedem anderen Registerinhalt addieren, eine Konstante zu jedem Registerinhalt, den „Top of a Stack" zu jedem Registerinhalt, einen Wert, der sich im Stack befindet, zu jedem Registerinhalt, einen Tabellenwert zu jedem Registerinhalt, einen Eingabewert von einer E/A-Einheit zu einem beliebigen Registerinhalt oder den Inhalt eines beliebigen Speicherplatzes zu einem beliebigen Regi-

sterinhalt. Die Reihenfolge läßt sich auch umkehren, man kann jeden Registerinhalt zu einem der oben genannten Beispiele addieren (Ausnahmen: Man kann nichts zu einer Konstante addieren und man kann die auf den Programmzähler bezogene Adressierungsart nicht zur Spezifizierung eines Bestimmungsortes benutzen.) Dazu kommt, daß man jeden dieser Befehle mit 8-, 16- oder 32-Bit-Daten benutzen kann.

Arithmetische Befehle

Folgende Typen arithmetischer Befehle sind verfügbar: Addition (ADD), Subtraktion (SUB) und Vergleich (CMP) sind allgemeine 2-Operanden-Befehle. ADDX und SUBX werden benutzt, um mit Zahlen, die länger als 32 Bit sind, zu bearbeiten (das Bedingungsbit X im MC68000 bietet eine ähnliche Funktion, wie ein Carry-Bit bei den meisten anderen Mikroprozessoren). Zwei Multiplikations- und Divisions-Befehle sind verfügbar: vorzeichenbehaftet (MULS und DIVS) für Befehle einfacher Genauigkeit und vorzeichenlos (MULU und DIVU) für Befehle höherer Genauigkeit.

Die Befehle für Negation (NEG) und Löschen (CLR) erfordern lediglich einen einzigen Operand, zur Negation von Werten höherer Genauigkeit kann man NEGX verwenden. Zur Mischung verschiedener Datenwortlängen verfügt der MC68000 über einen Befehl für Vorzeichenerweiterung (EXT), während TST (Test) zur Prüfung auf die Bedingungen positiv, negativ oder Null dient. Eine spezielle Instruktion ist der unteilbare „Test-and-Set"-Befehl (TAS), der die Software-Synchronisation in Multi-Prozessor-Operationen sicherstellt.

Eine Variation des ADD-Befehls ermöglicht es dem MC68000, eine Beschränkung zu überwinden, die andere Mikroprozessoren aufweisen. Die normale „1½"-Adreß-Konfiguration der meisten Prozessoren macht es schwierig, konstante Werte (immediate) mit anderen Dingen als Registern zu benutzen. Der MC68000 vermeidet dies mit der ADDI-Instruktion, die es erlaubt, einen konstanten Wert von der Länge Byte, Wort oder Langwort zu einem Operanden im Speicher zu addieren, wobei jede zulässige Adressierungsart benutzt werden kann, die auf eine Speicheradresse zielt.

Der MC68000 hat keine Inkrement- oder Dekrement-Befehle. Warum ist das so? Die Idee, die dahintersteht, ist die gleichartige Behandlung aller Befehle. Ein Inkrementierungsbefehl addiert die Zahl 1 zu einer gewissen Quantität und wird sehr häufig zum Durchgehen einer Tabelle von Byte-Werten benutzt. Der Programmer eines MC68000 möchte häufig 16- oder 32-Bit-Daten manipulieren, wobei allerdings Schritte von zwei oder vier innerhalb der Tabellenadresse erforderlich werden. Die Inkrement- und Dekrement-Befehle wurden daher generalisiert, um sie für alle Datengrößen brauchbar zu machen und trotzdem den Geschwindigkeitsvorteil für einen Befehl, der kein konstantes Argument aufnehmen muß, zu wahren. Man löste dieses Problem mit dem MC68000-Befehl „Schnelle Addition" (ADDQ) und „Schnelle Subtraktion" (SUBQ), mit denen eine Zahl zwischen 1 und 8 zu oder von jedem Register sowie jedem Speicherplatz addiert bzw. subtrahiert werden kann. Der Befehl führt dies in der kürzesten möglichen Zeit mit Hilfe von 3 Bits innerhalb eines 16-Bit-Op-Codes aus, durch die die Größe des Schrittes festgelegt wird. Auf diese Weise kann man hiermit sehr schnell und einfach einen Adreß-Pointer um 1, 2 oder 4 für 8-, 16- oder 32-Bit-Daten ändern, man kann aber auch Schritte von 3, 5, 6, 7 oder 8 realisieren. Der Effekt ist identisch mit dem Standard-Befehl ADDI, sogar die Codes im Statusregister sind gleich.

Zu den arithmetischen Befehlen gehören EXT, CLR und TEST. Weil drei verschiedene Datengrößen vom MC68000 verwendet werden, soll es einen einfachen Weg zur Veränderung dieser Größe geben. Wenn man lediglich einen Teil von einem Datum (z. B. die unteren 16 Bits eines 32-Bit-Registerinhaltes) verschieben möchte, benötigt man lediglich einen MOVE-Befehl für die richtige Datengröße. Wenn man allerdings ein Datum zu einem Zweier-Komplement mit größeren Abmessungen umwandeln will (z. B. wenn ein 16-Bit-Wert zu einem 32-Bit-Ausdruck gemacht werden soll) dann benötigt man einen Spezialbefehl. Der Befehl EXT dupliziert die oberen Bitpositionen von 8- und 16-Bit-Daten in die oberen Bereiche jedes Datenregisters, um das Datum auf 16 oder 32 Bit zu bringen. CLR lädt eine entsprechende Menge von Nullen in den Bestimmungsort. TST setzt die Negativ- und Null-Bedingungs-Bits entsprechend der Art des gegebenen Operanden.

Status-Register-Codes und Arithmetik erhöhter Genauigkeit

Was kann man tun, wenn man mit binären ganzzahligen Werten arbeitet und mehr als 32 Bit für seinen Ausdruck benötigt? Beispielsweise, wenn man zwei Zahlen mit 128 Bit (16 Byte) addieren möchte. Wenn beide Zahlen sich in den Registern des MC68000 befinden, sind alle acht Datenregister belegt. Häufiger befinden sich die beiden Zahlen in 16 aufeinanderfolgenden Bytes des Speichers, beginnend mit dem höchstwertigen Datenbyte. Das übliche Verfahren zur Addition solcher Zahlen ist die Addition der beiden niedrigwertigen Bytes, Zwischenspeichern des Übertrages, Addition der nächsthöheren Bytes, Zwischenspeichern des Übertrages usw. Diese Operationssequenz wird vom MC68000 durch eine vordekrementierte Adreß-Register-Betriebsart erledigt, die die Bezeichnung „-(An)" trägt. Jede Ausführung eines Befehls ADDX-(Am),-(An) dekrementiert die Werte in den Registern Am und An (m und n stehen für Zahlen zwischen 0 und 7), dann addiert er die zwei Zahlen, auf die von den Registerinhalten gezeigt wird. Wenn man diesen Befehl in einer Schleife ausführen läßt, kann man sehr schnell den Code erzeugen, der für die Berechnung von Zahlen mit hoher Genauigkeit benötigt wird.

Bild 7 zeigt das Status-Register des MC68000. Es enthält die Standard-Informationen Übertrag (C), Überlauf (V), Null (Z) und Negativ (N). Darüber hinaus findet man ein Status-Register-Bit, das andere Mikroprozessoren nicht kennen, nämlich das Bit X („Extend"). Dieses Bit wurde hinzugefügt, um die Probleme zu vermeiden, die mit der üblichen zu häufigen Anwendung des Übertrags-Bits verbunden sind.

Um das X-Bit näher erläutern zu können, muß zunächst das Übertrags-Bit beschrieben werden. Wie bereits erwähnt, wird das Übertrags-Bit sehr häufig zu oft verwendet. Es verändert sich z. B. beim Additions-Befehl, aber es wird auf zwei verschiedenen Wegen benutzt. Manchmal ist es für aufeinanderfolgende Additionen wichtig, manchmal für Abfragen im Programm, z. B. bei Verzweigungen, d. h. für die Programmsteuerung.

Der MC68000 verfügt für jeden dieser beiden Zwecke über ein Bit. Sowohl das Übertrags- als auch das X-Bit werden in Abhängigkeit von dem Ergebnis einer Addition geändert. Allerdings wird das Übertrags-Bit vom Mikroprozessor für Programm-Steuerzwecke verwendet, während das X-Bit speziell für Arithmetik-Operationen größerer Genauigkeit Verwendung findet. Für ADD, SUB, NEG und besondere Schiebe- und Rotations-Befehle werden sowohl Übertrags- als auch X-Bits aktualisiert. Andere Befehle, z. B. MOVE, AND, OR, TST, CLR, MUL und DIV verändern lediglich das Übertrags-Bit. Hiermit wird verhindert, daß unerwünschte Veränderungen des jeweiligen Bits auftreten können.

Wegen des X-Bits wird die übliche Operation „Addition mit Übertrag" beim MC68000 zum Befehl „ADDX", d. h. „Addition mit Erweiterungs-Bit". Dies ist aus folgendem Grund wichtig: Wenn man eine Arithmetik-Operation erhöhter Genauigkeit beginnt und ein Teilresultat erhält, wird das Erweiterungs-Bit unverändert beibehalten, auch wenn man die Addition zurückstellen muß, um Datenverschiebungen mit der Move-Instruktion auszuführen. Die Programmierung ist wesentlich einfacher, weil man die Statusregister-Codes nicht sichern muß, wenn ein Interrupt eine Operation erhöhter Genauigkeit unterbricht.

Ein weiterer Punkt zur Arithmetik erhöhter Genauigkeit soll hier erwähnt werden. Was bedeutet das Negativ-Bit, wenn man die Operation beendet hat? Es zeigt, wie sein Name sagt, an, daß das Resultat positiv oder negativ ist. Bei den meisten Mikroprozessoren zeigt das Zero-Bit lediglich, daß der höchstwertige Bereich des Resultats 0 ist, aber nicht, daß das ganze Ergebnis 0 ist. Die Arithmetikinstruktionen erhöhter Genauigkeit für den MC68000 sind so ausgelegt, daß das Null-Bit den Status des gesamten Resultates wiedergibt. Dies erfolgt, indem Befehle erhöhter Genauigkeit das Null-Bit zurücksetzen. Bei diesem Verfahren darf der Programmierer nicht vergessen, das Null-Bit vor dem Beginn der Operation zu setzen.

Ein weiterer wichtiger Punkt ist noch in Zusammenhang mit den Arithmetik-Operationen zu erwähnen, an dem sich ein weiterer grundsätzlicher Unterschied zwischen dem 68000 und vielen anderen Mikroprozessoren zeigt. Wie oft hat man, auch als erfahrener Programmierer, eine Serie von arithmetischen Operationen unterbrochen, um einige Speicher-Pointer zu modifizieren, und dabei später entdeckt, daß man für die Arithmetikoperation ein falsches Resultat erhält, weil man unbeabsichtigt die Status-Register-Code-Bits verändert hat. Viele Fehlermöglichkeiten verbergen sich hinter einer auf den ersten Blick klar erscheinenden Architektur.

Auch hier haben die Entwickler des 68000 die Probleme gelöst. Einer der wichtigsten Unterschiede zwischen Daten- und Adreß-Registern im MC68000 ist, daß Befehle, die ein Adreßregister als Bestimmungsort enthalten, nicht die Status-Register-Code-Bits verändern. Es kommen keine Modifikationen vor, wenn ein neuer Pointer-Wert in ein Adreßregister geladen wird, eine Inkrementierung oder Dekrementierung des Adreßregisters erfolgt, oder wenn man einen beliebigen Wert zu einem Adreß-Registerinhalt addiert. Dies bedeutet, daß die laufenden Datenoperationen von diesen Modifikationen nicht betroffen sind.

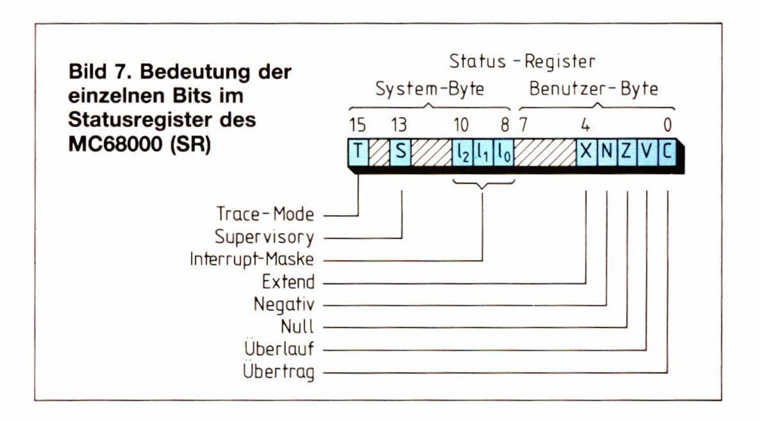


Bild 7. Bedeutung der einzelnen Bits im Statusregister des MC68000 (SR)

Ein anderer interessanter Punkt ist, daß alle Operationen für ein beliebiges Adreßregister das gesamte Register betreffen. Weil alle Adressen des MC68000 eine Breite von 32 Bit aufweisen, müssen alle Operationen mit einem Adreßregister als Bestimmungsort so ausgeführt werden, daß das Resultat eine 32-Bit-Adresse ist. Eine Lösung, bei der es erforderlich ist, daß alle Eingangswerte für Adreßregister-Operationen die volle Breite von 32 Bit annehmen, wäre eine Verschwendung von Speicherplatz. So werden entweder Wort- (16 Bit) oder Lang-Wort-Operationen (32 Bit) in den Adreßregistern A0...A7 ausgeführt. Wenn es sich um eine Wort-Operation handelt, werden die 16 Bit zunächst auf 32 Bit erweitert, bevor sie Verwendung finden.

Prozessorgeschwindigkeit

Wie schnell führt ein MC68000 die Instruktionen aus? Aufgrund des Aufbaues eines Mikroprozessors lassen sich die Additionsbefehle als Richtlinie für alle Arithmetik- und Logik-Instruktionen heranziehen. Ein Prefet-

ching-Mechanismus im MC68000 stellt die decodierten Instruktionen zur Verfügung, die ausgeführt werden müssen. Während die Zeitinformation lediglich angibt, wie lange es dauert, bis der Additionsvorgang ausgeführt ist, muß man berücksichtigen, daß der Prefetcher bereits den nächsten Op-Code aufnimmt, während der vorliegende Op-Code ausgeführt wird.

Die kürzeste Zeit, die der Mikroprozessor MC68000 zum Zugriff auf einen Speicher (Schreiben oder Lesen) braucht, dauert 4 Taktzyklen. Bei einer Taktfrequenz von 8 MHz (Standard-Version des MC68000) nimmt dieser Buszyklus 500 ns in Anspruch. (Alle folgenden Zeitangaben werden in Taktzyklen gemacht, weil die Mikroprozessorfamilie MC68000 aus Versionen mit 8, 10 oder 12,5 MHz besteht.) Jeder Befehl nimmt wenigstens vier Taktzyklen in Anspruch, weil dies die Zeit ist, um den nächsten Op-Code aufzunehmen.

Der MC68000 hat lediglich eine 16-Bit-ALU für Datenoperationen. Daher lassen sich 8- oder 16-Bit-Operationen in einem einzigen Durchlauf verarbeiten; dieser erfordert vier Taktzyklen. Eine 32-Bit-Operation erfordert einen zweiten Durchlauf. Speicher-Adressierarten verlängern die erforderliche Zeit, weil der Mikroprozessor mehr Zeit zur Berechnung der Adressen benötigt und ein Buszyklus für jeweils 16 Bit Adressierungsinformation oder Daten, die zu transferieren sind, erforderlich wird. Eine Index-Adressierart oder ein Vorgang mit Displacement z. B. erfordert einen zusätzlichen Buszyklus für das Adreß-Erweiterungswort und einen weiteren, um die Daten zu erhalten (zwei, wenn es sich bei den Daten um ein Langwort handelt); Addition etwa acht zusätzliche Taktzyklen (12 wenn es sich bei den Daten um ein Langwort handelt) zu der Ausführungszeit einer gegebenen Instruktion, die in dieser Betriebsart abläuft. Einige Beispiele für die längsten Ausführungszeiten verschiedener Additionsbefehle zeigt *Tabelle 3.*

Tabelle 3. Beispiele für Additionsbefehle des 68000

Befehl	Operation
ADD.B D6,D2	Addition der unteren 8 Bit von D6 und D2 (4 Taktzyklen)
ADD.L 52(A1, D7.W),D6	Die effektive Adresse ist die Summe der Konstante 52, den Inhaltes von Reg-A1 und die unteren 16 Bit von Reg. D7. Das Langwort der effektiven Adresse wird zum Inhalt des Reg. D6 addiert (20 Taktzyklen)
ADD.W D3,(A7)	Addiert die unteren 16 Bit von D3 zum Element, auf das der Top of Stack in A7 zeigt (12 Taktzyklen)
ADDI.L#$400,D1	Addiert 4004 Hex zum 32-Bit-Inhalt von D1 (16 Taktzyklen)
ADDI.B#$A9, $30B(A6)	Die effektive Bestimmungsadresse ist die Summe von 30BH und dem Inhalt von Register A6; H9Hex wird zum Byte an der effektiven Adresse addiert (20 Taktzyklen)
ADDA.W−(A5),A2	Dekrement A5 und 2, dann das Wort, auf das Wort in A5 zeigt zu Register A2 addieren (14 Taktzyklen)
ADDA.W#100,A5	Addiere den Wert 100 zum Inhalt des Registers A5 (12 Taktzyklen)
ADDQ.W#1,(A4)+	Addiere 1 zum Wort, auf das A4 zeigt, dann inkrementiere Register A4 und 2 (12 Taktzyklen)
ADDQ.B#3,D7	Addiere 3 zum Inhalt von Register D7 (4 Taktzyklen)
ADDX.L−(A2), −(A5)	Nach Dekrementierung der Register A2 und A5 um 4 addiere das X-Bit und das Langwort, auf das A2 und A5 zeigen (30 Taktzyklen)

Wie die ADD-Instruktion liegen die Arithmetik-Befehle des MC68000 in unterschiedlicher Form vor. Die Subtraktionsbefehle haben ähnliche Versionen wie die Additionsbefehle: SUB, SUBA, SUBI, SUBQ und SUBX. Befehle für Vergleichsoperationen sind ebenfalls ähnlich (CMP, CMPA, CMPI). Sie führen die Subtraktionen ohne das Speichern des Ergebnisses aus, als Information werden lediglich die entsprechenden Statusregister-Bits gesetzt. Eine Speicher-Vergleichsinstruktion (CMPM) erlaubt es, daß zwei Zeichenketten aus binären ganzen Zahlen im Speicher miteinander verglichen werden. Es gibt zwei Versionen der Ein-Operanden-Negations-Instruktion: NEG und NEGX, wobei der Zustand des X-Bit jeweils ignoriert oder miteinbezogen wird.

Multiplikation und Division

Zwei Versionen von Multiplikations- und Divisions-Befehlen bringen den Nutzen einer komplexeren Arithmetik. Die beiden Versionen sind ohne Vorzeichen (MULU und DIVU) sowie mit Vorzeichen (MULS und DIVS). Diese Versionen interpretieren ihre Operanden als Einerkomplement und Zweierkomplement. Alle diese Befehle können Immediate-Werte als Multiplizierer oder Divisior enthalten, so daß auch mit Konstanten gerechnet werden kann.

Die Multiplikationsbefehle benötigen zwei 16-Bit-Operanden (einen von irgendeinem Speicherplatz, der durch die Adressierart festgelegt ist oder aus irgendeinem Datenregister, der andere von den unteren 16 Bit jedes Datenregisters) multiplizieren diese und geben das resultierende Produkt in die 32 Stellen des gleichen Datenregisters. Die Divisions-Befehle nehmen den Dividenden von irgendeinem 32-Bit-Datenregister und teilen diesen durch einen 16-Bit-Divisor, der aus dem Speicher stammen kann, wobei jede beliebige Adressierart zur Anwendung kommt, oder aus jedem Datenregister. Der Quotient befindet sich in den unteren 16 Bit des gleichen 32-Bit-Datenregisters, während der 16-Bit-Rest die oberen 16 Bit des gleichen Registers ausfüllt.

Der Divisionsbefehl hat zwei Eigenschaften, die unerwünscht sein können und daher besonderer Behandlung bedürfen. Eine Divison durch Null ergibt die Zahl Unendlich, wenn sie überhaupt definiert ist. Um Fehler zu verhindern, verfügt der 68000 über eine Schaltung, die dafür sorgt, daß eine Division durch Null nicht ausgeführt wird.

Der andere Fall, der eintreten kann, ist, daß die Division zu klein für den Dividenden ist und der Quotient mehr als 16 Bit erforderlich macht. Wenn diese Überlaufbedingung erkannt wird, hält die Division an, das Überlauf-Statusregister-Bit (V) ist gesetzt und der

Input: Register D0 enthält 32-Bit-Multiplikand
Register D1 enthält 32-Bit-Multiplikator

Output: Register D0 und D1 enthalten das 64-Bit-Resultat, (MSB in D0)

```
        SUBQ     #4,A7        initialize product area
        CLR.L    -(A7)
        MOVE.L   D0,-(A7)     save copy of multiplicand

        MULU     D1,D0        multiply low-order parts
        MOVE.L   D0,8(A7)

        MOVE.W   (A7),D0      high-order multiplicand
        MULU     D1,D0        times low-order multiplier
        ADD.L    D0,6(A7)

        SWAP     D1           now use high-order multiplier
        MOVE.W   2(A7),D0     low-order multiplicand
        MULU     D1,D0        times high-order multiplier
        ADD.L    D0,6(A7)
        BCC      MUL32A       carry into high-order
        ADDQ.W   #1,4(A7)     word of product

MUL32A  MOVE.L   (A7)+,D0     high-order multiplicand
        SWAP     D0
        MULU     D1,D0        times high-order multiplier
        ADD.L    (A7)+,D0
        ADDQ     #4,A7
        MOVE.L   (A7)+,D1     load low-order product
```

Bild 8. Dieses kurze Assemblersprachenprogramm multipliziert zwei 32-Bit-Zahlen

Befehl wird zu Ende ausgeführt, ohne daß einer der ursprünglichen Operanden überschrieben wird. Man sollte daher nach jeder Divisionsoperation das Überlaufbit prüfen.

Aus mehreren Gründen gibt es keinen Befehl zur Multiplikation von zwei Zahlen mit 32 Bit oder zur Division einer 64-Bit-Zahl durch eine 32-Bit-Zahl. Erstens werden diese Befehle sehr selten in der Praxis benötigt, zweitens gibt es keine Einrichtungen innerhalb der Maschine, um Worte mit 64 Bit Breite zu behandeln. Weil schließlich solche Instruktionen den Prozessor sehr lange beschäftigt halten, würde man beim MC68000 auch wesentlich länger auf eine Interrupt-Antwort warten. Diese Situation ist nicht gewünscht.

Die Multiplikationsbefehle nehmen weniger als 17 Taktzyklen zur Ausführung mit Registeroperanden in Anspruch, die Divisionsbefehle erfordern weniger als 140 Taktzyklen für eine Operation ohne Vorzeichen (158 mit Vorzeichen). Eine kurze MC68000-Routine, die eine 32×32-Bit-Multiplikation ausführt, zeigt ein Listing in *Bild 8*. Diese läuft in etwa 60 µs ab.

BCD-Arithmetik

Der letzte Typ der Arithmetikbefehle verarbeitet dezimale Ziffern. Am häufigsten findet man bei der Interaktion zwischen Mensch und Maschine BCD-Daten. Bei dieser Codierung wird Zahleninformation als eine Kette von Bits dargestellt, die jede dezimale Zahl in eine 4-Bit-Binärzahl umwandelt. Zahlen lassen sich sehr einfach in BCD-Form umsetzen; wenn sie sich erst einmal im Computer befinden, sind sie leicht in einer Form zu drucken, die vom Menschen lesbar ist. Weil das BCD-Format so nützlich ist, besitzen die meisten Mikroprozessoren Befehle, die direkt mit BCD-Zahlen arbeiten. Zur Manipulation von BCD-Daten verfügt der MC68000 über drei Befehle zur Addition (ABCD), Subtraktion (FBCD) und Negation (NBCD). Jede dieser Instruktionen arbeitet mit zwei BCD-Ziffern, die in einem Byte untergebracht sind.

Weil BCD-Zahlen mehrere Stellen umfassen können, arbeiten BCD-Befehle als Operationen erhöhter Genauigkeit, was bedeutet, daß sie die Eigenschaften anderer Instruktionen erhöhter Genauigkeit haben.

Die Operanden können in Datenregistern oder im Speicher sein. Der Wert des X-Status-Register-Bits ist in den BCD-Operationen enthalten und das Z-Statusbit wird so behandelt, daß es den Zustand des gesamten Resultats und nicht nur eines Teils davon wiedergibt.

Auch hier ist die beste Eigenschaft der Befehle die Einfachheit, mit der sie arbeiten, insbesondere, wenn sie mit dem oftmals mysteriösen Code älterer Mikroprozessoren verglichen wird. Einen Eindruck vom MC68000-Code, der BCD-Funktionen ausführt, erhält man in *Bild 9*. Hier werden zwei 6stellige Zahlen addiert.

Die drei ABCD-Befehle (Add binary coded-decimal) beginnen bei den niedrigwertigen beiden Ziffern und gehen bis zum höchstwertigen vor. Dies ist erforderlich, um die richtigen Resultate unter Verwendung des Erweiterungsbits zu erhalten. Das Ergebnis ersetzt die BCD-Zahl, auf die von A2 gezeigt wird. Wenn die Routine beendet ist, zeigt A2 auf das erste Byte des BCD-Resultates. Ähnlich arbeiten die Operationen für Subtraktion und Negation.

Logik-Befehle

Die Logik-Befehle des MC68000 sind einfach, aber leistungsfähig. Es handelt sich um die Befehle AND, OR, EOR und NOT. Wie die arithmetischen Befehle sind 8-, 16- und 32-Bit-Worte zur Verarbeitung möglich, die sich in den Datenregistern oder im Speicherbereich befinden. Diese Befehle sind ebenso schnell wie die arithmetischen Instruktionen. Darüber hinaus lassen sich die Instruktionen ANDI, ORE und EORE zum Setzen, Rücksetzen und Umschalten einzelner Statusregister-Bits verwenden.

Ein serieller Shifter im MC68000 kann jede Anzahl von Bits (8, 16 oder 32) verschieben. Der Befehl ASR (arithmetischer Rechts-Schiebevorgang) schiebt das niedrigwertige Bit bis zu den X- und C-Statusbits, während das höchstwertige Bit vor dem Weiterschieben nach rechts dupliziert wird. Beim Befehl ASL (arithmetischer Links-Schiebevorgang), dem logischen Rechts-Schieben (LSR) und dem logischen Links-Schieben

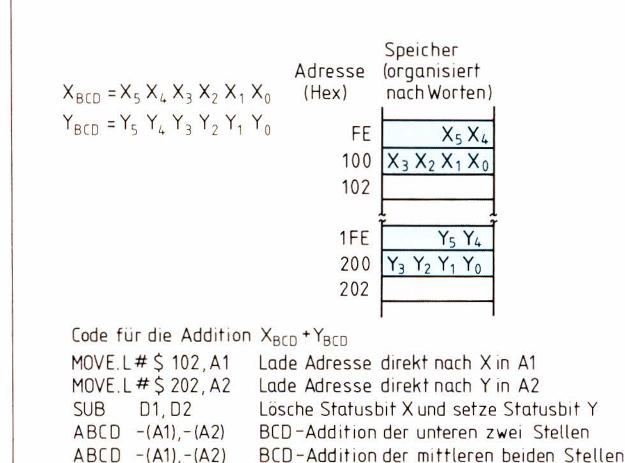


Bild 9. Beispiel für BCD-Arithmetik erhöhter Genauigkeit. Wegen der Adressierungsart mit Vordekrementierung („ABCD-(A1),-(A2)") sind die Register-Pointer schon vor der BCD-Addition dekrementiert. Daher müssen die Register A1 und A2 mit einem Wert geladen werden, der auf das Byte zeigt, das direkt hinter dem LSB der zu verarbeitenden Zahl liegt

(LSL) gehen die herausgeschobenen Daten in die X- und C-Bits, während die leeren Stellen mit Nullen gefüllt werden.

Die Rotationsbefehle schieben die Bits kreisförmig herum, so daß die Bits, die an einem Ende des Operanden herausfallen, am anderen Ende wieder hineingeschoben werden. Dies bewirkt, daß ein Bit, das aus dem Datenbereich herausgeschoben wurde, auch in das Codierungsbit des C-Statusregisters eingeschoben wird bzw. in das X-Bit. Die Rotationsbefehle sorgen für Rotation in Rechts- und Links-Richtung (ROR und ROL). Die Befehle ROXR und ROXL werden benutzt, wenn man sowohl das X- als auch das C-Bit aktualisieren möchte.

Ein einziger Schiebe- und Rotations-Befehl kann Registerdaten nicht weniger als 32 Bitpositionen in der gewählten Richtung verschieben. Man kann die Anzahl der Stellen entweder statisch spezifizieren (ein Wert zwischen 1 und 8), der im Befehls-Opcode codiert ist, wenn der Befehl geschrieben wird, bzw. dynamisch (als Wert zwischen 0 und 63, der in einem spezifizierten Datenregister abgelegt ist), wenn der Befehl ausgeführt wird. Zur Vereinfachung sind die Speicheroperanden, die verschoben oder rotiert werden sollen, auf Displacements von einem Bit und Operationen mit Daten von Wortlänge beschränkt. *Tabelle 4* zeigt einige Schiebe- und Rotationsbefehle, deren zeitliche Abläufe und die praktische Wirkung.

Wichtig, insbesondere für E/A-Operationen, sind 1-Bit-Manipulationen. Bisher benutzte man zu ihrer Realisierung UND-, ODER- sowie EOR-Befehle. Der Nachteil solcher Operationen ist allerdings, daß sie relativ primitiv sind. Die leistungsfähige CPU MC68000 verfügt über Befehle, durch die die Bit-Manipulation wesentlich einfacher zu realisieren ist. Es handelt sich um die Befehle BTST (Bit-Test), BSET (Bit-Testen und -Setzen), BCLR (Bit testen und löschen) sowie BCHG (Bit testen und verändern). Wie kann man das Ziel-Bit spezifizieren? Beim MC68000 kommen dazu zwei Methoden zur Anwendung, die denen für die Schiebe- und Rotier-

Tabelle 4. Beispiele für Schiebe- und Rotierbefehle

Status-Register-Instruktion	Register						Status-Register-Code		
							C'	X	V
ASR.B#3,D3	(D3 vorher)	10111010	01011111	01100101	10101100		X	X	X
	(D3 nachher)	10111010	01011111	01100101	11110101	(12 Taktzyklen)	1	1	0
ASL.L#5,D1	(D1 vorher)	11101100	10100010	11011101	00101111		X	X	X
	(D1 nachher)	10010100	01011011	10100101	11100000	(18 Taktzyklen)	1	1	1
LSL.W D5,D7	(D5 vorher)	00101000	10001100	11101001	00101001		X	X	X
	(D7 vorher)	10111010	01011111	01100101	00010101		X	X	X
	(D7 nachher)	10111010	01011111	00101010	00000000	(24 Taktzyklen)	0	0	0
ROL.L D2,D1	(D2 vorher)	01100101	00101010	10111110	01110100		X	X	X
	(D1 vorher)	10010101	00101000	01000101	10010100		X	X	X
	(D1 nachher)	01011001	01001001	01010010	10000100	(48 Taktzyklen)	0	X	0
ROXR.W #4,D6	(D6 vorher)	10111010	01011111	01100101	00010101		X	P	X
	(D6 nachher)	10111010	01011111	101P0110	01010001	(14 Taktzyklen)	0	0	0
ROR $A0000	(Wort A0000 vorher)		10011100	10101001			X	X	X
	(Wort A0000 nachher)		11001110	01010100			1	X	0

Erklärung:
„X" bedeutet entweder „1" oder „0"

Befehle ähnlich sind. Entweder gibt ein Datenregister oder eine Serie von Bits im Opcode des Bit-Befehls an, welches Bit betroffen ist. Wenn ein Register benutzt wird, kann die Bit-Nummer zwischen 0 und 31 variieren, oder von 0...7, wenn es sich um einen Speicherplatz handelt. (Beim MC68000 werden die Bits im Speicher durch die Bit-Nummer des Bytes, in dem sie sich befinden, identifiziert.)

Mit echten Bit-Manipulationsbefehlen sind viele Operationen wesentlich einfacher realisierbar, z. B. die Abfrage des Zustandes von Eingängen, Ausgängen, das Setzen von Registerbits, von Attributbits, Umsetzen von Bit-Matritzen oder der Aufbau spezieller Datentypen.

3 Befehle hoher Leistung: Verzweigungen und Sprünge

Datenverschiebung, arithmetische und logische Befehle führen den größten Teil der Rechenarbeit in Programmen aus, bei Computern handelt es sich allerdings um Geräte, die mehr können als einfache Addiermaschinen. Es sind daher Befehle zur Programmsteuerung notwendig. Diese Instruktionen geben Computern die Möglichkeiten, Entscheidungen herbeizuführen, indem nicht direkt aufeinander folgende Bereiche des Codes in Abhängigkeit bestimmter, im Programmverlauf auftretender, Bedingungen abgearbeitet werden. Verzweigungsbefehle sorgen dafür, daß Programmteile abgearbeitet werden, die sich unter einer effektiven Adresse befinden, die die Summe des laufenden Inhaltes vom Programmzähler und einem vorgegebenen Offset darstellt. Verzweigungsbefehle benutzt man insbesondere in den Fällen, wenn man positionsunabhängigen Code schreibt. Sprungbefehle unterscheiden sich von Verzweigungsbefehlen darin, daß die Sprungbefehle sich auf absolute Speicheradressen beziehen, keine Bedingung erfordern und jede Adressierart des MC68000 zur Spezifizierung des Zielortes benutzen können.

Die CPU MC68000 verfügt über eine flexible bedingte Verzweigungsinstruktion, die die Bezeichnung „Bcc" trägt, wobei die Buchstaben cc auf die unterschiedlichen Bedingungen hinweisen, die sich spezifizieren lassen. Es gibt 14 verschiedene Bedingungen, z. B. solche Dinge die größer als (BGT), kleiner oder gleich (BLE), gleich (BEQ), Überlauf (BVS) und niedriger oder gleich (BLS); eine vollständige Übersicht gibt *Tabelle* 5. Der Befehl BRA ist nicht bedingt, führt beim Auftreten allerdings immer zur Verzweigung. Verzweigung erfolgt durch die Addition eines bestimmten Wertes zum Programmzähler. Alle Verzweigungsbefehle umfassen einen 8-Bit- oder 16-Bit-Displacement-Wert mit Vorzeichen, der zu addieren ist. Weil dieser Wert ein Vorzeichen besitzt, ist sowohl eine Vorwärts- als eine Rückwärts-Verzweigung möglich.

Obwohl alle Befehle des MC68000 einen Umfang haben, der ein Mehrfaches von 16 Bit darstellt und entsprechend der Wortgrenzen normiert sein müssen, interpretiert der MC68000 das Displacement in allen Verzweigungsoperationen als Byte und nicht als Wort. Dies hat seinen Grund darin, daß man der Maschine ein möglichst großes Maß an Flexibilität geben möchte, gleichzeitig aber alle Möglichkeiten für zukünftige Erweiterungen offen hält. Die Begrenzung der Maschine auf Wort-Offsets würde dafür sorgen, daß zukünftige Mitglieder der MC68000-Familie Befehle haben, die nicht an einer Wortgrenze beginnen oder Mehrfaches von 8 Bit sind. Ein 16-Bit-Offset ergibt einen Adreßbereich von −32 768 Byte...+32 767 Byte, während die bisherigen 8-Bit-Grenzen herkömmlicher Computer lediglich eine Adreßvariation von −128...+127 Byte zulassen.

Spezielle Versionen der Sprung- und Verzweigungsbefehle existieren außerdem für Unterprogramm-Aufrufe. Man kann in ein Unterprogramm verzweigen (BSR) indem man einen Displacement-Wert benutzt, oder man kann in das Unterprogramm einspringen (JSR), indem man die absolute Adresse spezifiziert. Unterprogrammaufrufe sorgen dafür, daß die Rückkehradresse (der laufende Wert des Programmzählers) im Systemstack gesichert wird, bevor die Steuerung an die Unterprogramm-Routine übergeht. Die Rückkehradresse wird vom Stack zurückgeholt und zum Programmzähler gebracht, wenn der MC68000 den Befehl RTS (Return from Subroutine) ausführt.

Manchmal muß man auch die Condition-Codes sichern, die vor dem Aufruf eines Unterprogramms existierten. Dies erfolgt sehr einfach mit Hilfe des Befehls MOVE SR,-(A7). Hiermit wird der Inhalt des Statusregisters in den System-Stack, auf den Register A7 zeigt, gebracht. Man kann auch mit nur einem einzigen MOVEM-Befehl den Inhalt angewählter Register sichern. Am Ende des Unterprogramms läßt sich wiederum ein MOVEM-Befehl zum Wiederherstellen des alten Zustandes benutzen; danach folgt der Befehl RTR, der dafür sorgt, daß die abgelegten Condition-Codes zurückgeholt und ihre Speicherplätze gebracht werden.

Tabelle 5. Bedingungstest für die Befehlsgruppen B_{cc} und DB_{cc}

Mnemonic	Beschreibung der Bedingung	getestete Flags
T	true	1
F	false	0
HI	high	$\overline{C} \wedge \overline{Z}$
LS	low or same	$C \underline{+} Z$
CC	carry clear	$\overline{C}$
CS	carry set	C
NE	not equal	$\overline{Z}$
EQ	equal	Z
VC	overflow clear	$\overline{V}$
VS	overflow set	V
PL	plus	$\overline{N}$
MI	minus	N
GE	greater or equal	$(N \wedge V) \vee (\overline{N} \wedge \overline{V})$
LT	less than	$(N \wedge \overline{V}) \vee (\overline{N} \wedge V)$
GT	greater than	$(N \wedge V \wedge \overline{Z}) \vee (\overline{N} \wedge \overline{V} \wedge \overline{Z})$
LE	less or equal	$Z \vee (N \wedge \overline{V}) \vee (N \wedge V)$

Schleifen und Ketten

Sehr häufig wird eine Rückwärtsverzweigung dazu benutzt, eine Programmschleife aufzubauen, die einen wichtigen Teil der Programmierung darstellt, weil mit ihr Operationen möglich sind, die so lange wiederholt werden, bis ein gewisser Zustand oder eine Bedingung erreicht ist. Schleifen können entweder unter einer gewissen Bedingung oder bei Erreichen einer gewissen Zahl von Schritten beendet werden. Eine Schleife, die von beiden Methoden gesteuert wird, ist in vielen Fällen sehr nützlich. Die doppelte Bedingung ermöglicht es, daß eine Schleife so lange durchlaufen wird, bis eine vorgegebene Bedingung erreicht ist, während sichergestellt ist, daß die Schleife keine ungültigen Daten verarbeitet oder unendlich lange abläuft.

```
     MOVEQ   #$13,D4         Lade Abschlußzeichen in Register
     MOVE    #COUNT,D3       Lade String-Länge
     MOVE.L    STRING,A3     Lade String-Anfang
     LEA     *+TABLE,A2      Offset für Umsetztabelle
     CLR     D0              Vorbereitung Index
     BRA     POOL            Start Umsetzung
LOOP MOVE.B  0(A2,D0),(A1)+  Umsetzung und Speichern der Ergebnisse
POOL MOVE.B  (A1),D0         Lade nächstes Zeichen
     CMP.B   D4,D0           Abschlußzeichen gefunden ?
     DBEQ    D3,LOOP         Wenn nicht und kein Zeichenkastenende,
                             dann Verzweigung

     Bei der Umsetzung von n Bytes sind
     72+(40×n) Takte erforderlich
     (649 µs für 128 Byte bei 8 MHz)
```

Bild 10. Beispiel für ein String-Translation-Programm, das den Befehl DBEQ benutzt, um eine Schleife unter zwei Bedingungen zu beenden, nämlich entweder bei „End of String" (definiert durch die Zeichenheftenlänge in D4) oder beim Vorliegen eines Abschlußzeichens (auch im Register D4). Dieses Programm setzt eine Zeichenkette zeichenweise entsprechend der in der „TABLE" gespeicherten Wert zusammen. Bei einem bestimmten Zeichen wird der Wert (gespeichert im Register D0) als ein Index für die „TABLE" (A2 zeigt darauf) benutzt. Die eigentliche Umsetzung erfolgt in „Loop"

Der MC68000 verfügt über eine passende Instruktion mit der Bezeichnung „DBcc" (Decrement Counter and Branch conditionally). Diese Instruktion benutzt jedes Datenregister als Zähler und verzweigt sowohl bei Erfüllung einer bestimmten Bedingung und bei Erreichen des spezifizierten Wertes eines Datenregisters. Folgende Ereignisse finden nacheinander statt: Zuerst prüft der MC68000, ob die festgelegte Bedingung erfüllt ist; wenn das der Fall ist, beendet er die Schleife. Wenn die Bedingung nicht erfüllt ist, wird das spezifizierte Register um eine 1 dekrementiert. Falls sich der Wert -1 ergibt, wird die Schleife ebenfalls beendet, im anderen Fall erfolgt eine Verzweigung zum Anfang der Schleife.

Es ist zu beachten, daß der Befehl DBcc die Register auf den Wert -1 prüft. Dafür gibt es einen wichtigen Grund: Die meisten Schleifen erfordern zusätzliche Schritte, um sicherzustellen, daß die Schleife null mal ausgeführt werden kann, wenn dies erforderlich ist, und daß die Schleife die gewünschte Bedingung testet, bevor ein Schritt ausgeführt wird. Der DBcc-Befehl ist so aufgebaut, daß sich beide hier genannten Bedingungen erfüllen lassen, ohne daß ein zweiter Test erforderlich wird. Zusätzlich läßt sich mit einem einfachen bedingten Sprungbefehl feststellen, ob das Programm die Schleife verlassen hat, weil die Anzahl der Schritte erreicht war oder die Bedingung erfüllt ist.

Der Befehl DBcc erlaubt über eine große Menge von String-Operationen, insbesondere in Verbindung mit Vor-Dekrement- und Nach-Inkrement-Adressierarten. Mit Hilfe der entsprechenden MOVE-Befehle, z. B.

MOVE Dn,(An)+;
MOVE (An)+,(An)+;
MOVE−(An),−(An);
MOVE (An)+,−(An)

auf die ein DBcc-Befehl folgt, kann man dafür sorgen, daß der MC68000 einen Speicherblock füllt, Zeichenketten kopiert oder umkehrt.

CMPM−(An),−(An)

vergleicht zusammen mit DBNE zwei Zeichenketten, während

CMP Dn,−(An)

zusammen mit DBEQ eine Zeichenkette daraufhin untersucht, ob ein Muster übereinstimmt. Mehr-Befehls-Schleifen ergeben sehr leistungsfähige Zeichenketten-Operationen.

Man erkennt die Vorteile der DBcc-Instruktion in einem Assemblersprachenprogramm am besten anhand eines Beispieles, wie es in Bild 1 dargestellt ist. Hier wird eine Zeichenkette solange umgesetzt, bis das abschließende Zeichen auftritt oder das Ende der Kette erreicht ist. Register D3 beinhaltet die Kettenlänge, während Register D4 das letzte Zeichen enthält. Register A1 zeigt auf die Zeichenkette, während die Umsetztabelle unter einer Adresse zu finden ist, die in Register A2 untergebracht ist. Die hier dargestellte Routine läuft sehr schnell ab und zeigt, welche Leistung sich aus der Kombination vielseitiger Befehle und unterschiedlicher Adressierungsarten ergibt.

Unterstützung höherer Sprachen

Viele höhere Programmiersprachen, z. B. Pascal, benutzen anspruchsvolle Programmierkonzepte, die sich noch mit Hilfe sogenannter Reentrant- und Rekursiv-Programmierung sowie von Subroutinen mit lokalen Variablenbereichen verbessern lassen. Der MC68000 verfügt über die Einrichtungen zur Unterstützung dieser Techniken.

Man kann Reentrant-Codes mit verschiedenen Prozessen jederzeit eingeben, wobei diese jederzeit korrekte Resultate erhalten. Dieser ist sehr wichtig bei Interrupt-Routinen, die sich vor ihrem Abschluß selbst unterbrechen könnten. Nur Reentrant-Code kann die Interrupt-Routine beim zweiten Mal richtig ausführen, dann zur

unterbrochenen Version zurückkehren und diese richtig ausführen.

Rekursive Programme sind diejenigen, die sich selbst aufrufen. Beispiel dafür ist ein Programm, das eine direkte Linie zwischen zwei Punkten zeichnen soll, wobei jeweils der Mittelpunkt zwischen diesen zwei Punkten gezeichnet wird, worauf das Programm sich selbst aufruft, um mit den beiden Liniensegmenten in gleicher Weise zu arbeiten. Rekursive Programme dienen dazu, komplexe Algorithmen abzuarbeiten, wobei relativ kleine Mengen von Code erforderlich sind. Ihr Nachteil ist die langsame Ausführung und die starke Belastung des Stack-Bereiches, weil die Zwischenvariablen abgelegt werden müssen. Beim MC68000 findet man spezielle Befehle, die diese Aufgabe erleichtern, nämlich LINK und ULNK.

LINK und ULNK erlauben es Unterprogrammen, einen Teil des Stacks zum Speichern lokaler Variabler zu reservieren. Oft kommt es vor, daß ein Programmierer den Stack-Pointer dekrementiert, um einen Bereich des Speichers zu reservieren, wobei die Adresse des oberen Ende dieses Bereichs als Bezugspunkt dient. Diese Adresse wird auch „Frame-Pointer" (FP) genannt und ist ein Wert, der beim MC68000 in einem der sieben Adreßregister A0...A6 gespeichert ist. Der Stack-Pointer läuft beim Ausführen eines Unterprogramms auf und ab, wenn Stack-Operationen auszuführen sind. Ein stabiler Frame-Pointer stellt einen sehr guten Bezugspunkt für die Variablen dar, während der Stack-Pointer eine in weiten Bereichen variable Referenz zu den gleichen Variablen darstellt.

Um zu zeigen, wie LINK und ULNK dem Programmierer beim Zugriff auf lokale Variablenbereich helfen, sollte man sich *Bild 11* anschauen. Hier wird angenommen, daß man sich im Unterprogramm A befindet, das über einen eigenen lokalen Variablenbereich verfügt, auf den ein Frame-Pointer zeigt. Bevor Unterprogramm A das Unterprogramm B aufruft, setzt es zunächst Parameter an das Oberende des Stacks (Bild 11*a*). Nach dem Unterprogrammaufruf von B wird die Rücksprungadresse von A in den Stack geholt (Bild 11*b*). Die LINK-Instruktion enthält den Namen eines Adreßregisters, das als Frame-Register genommen wird sowie ein Displacement, das den Umfang des Speichers anzeigt, der für die lokalen Variablen freizuhalten ist. Wenn dies ausgeführt ist, geschehen drei Dinge (Bild 11*c*...11*e*): Der Inhalt des Frame-Pointers (zeigt auf einen Stack, der den vorhergehenden Frame-Pointer enthält) wird auf den Stack transferiert, der Frame-Pointer selbst wird mit dem Stack-Pointer identisch gemacht, und der Stack-Pointer wird mit Hilfe des in der Instruktion gegebenen Displacement verändert. Wie aus Bild 11e zu erkennen ist, zeigt der Stack-Pointer auf das obere Ende des Stacks und der Frame-Pointer zeigt auf ein Wort unter dem Bereich für die lokalen Variablen von Unterprogramm B. Wenn der Befehl ULNK ausgeführt wird, erfolgt dieser Prozeß in umgekehrter Reihenfolge (Bild 11*f*), wobei Unterprogramm B zur Ausführung einer RTS-Instruktion vorbereitet und damit die Steuerung an Unterprogramm A zurückgegeben wird.

Adreßberechnung mittels Hardware

Die meisten Mikroprozessor-Operationen haben entweder mit Daten oder mit der Programmsteuerung zu tun. Die meisten benutzten Speicheradressierarten verfü-

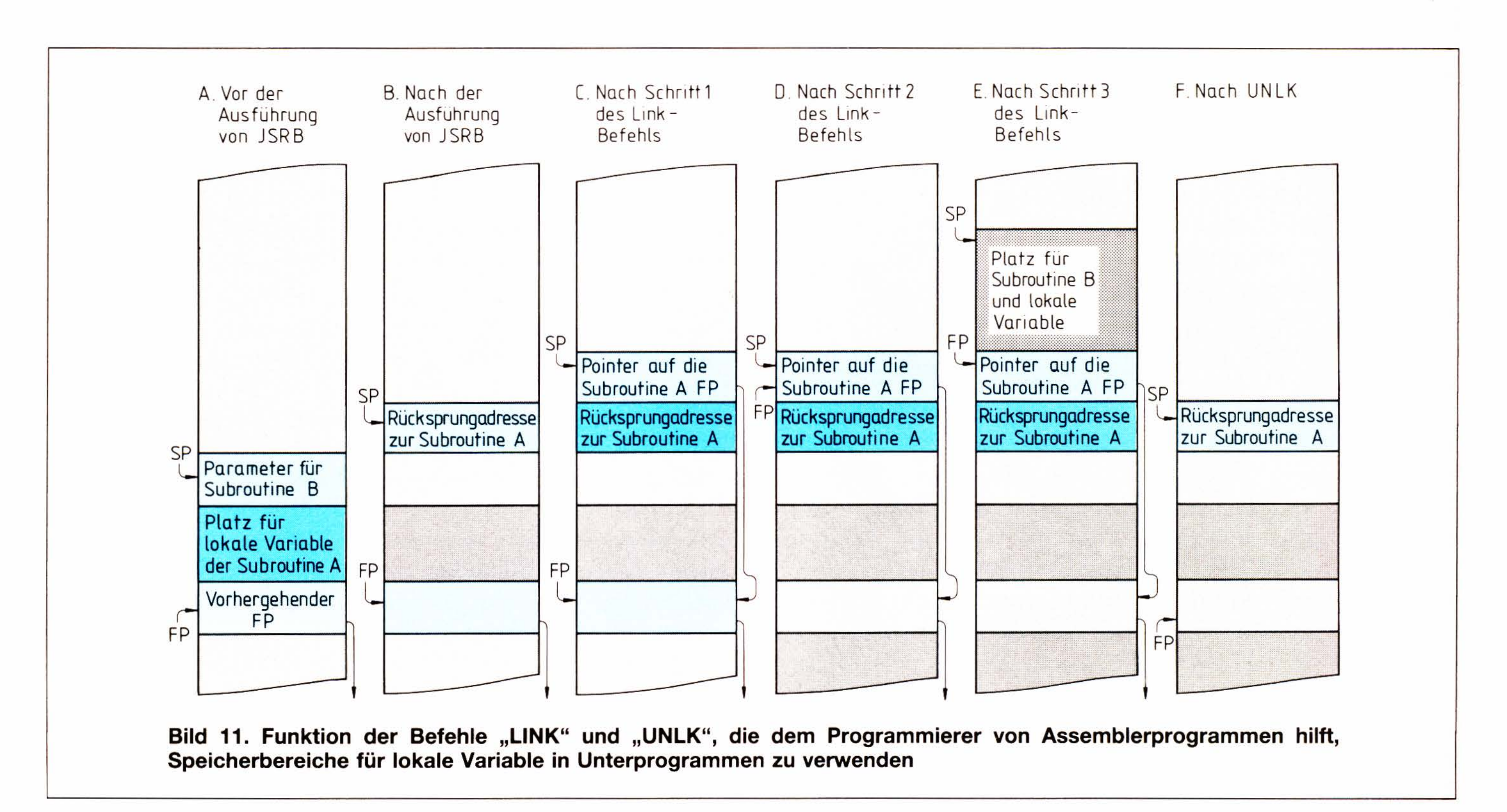


Bild 11. Funktion der Befehle „LINK" und „UNLK", die dem Programmierer von Assemblerprogrammen hilft, Speicherbereiche für lokale Variable in Unterprogrammen zu verwenden

gen über unterschiedliche Methoden zur Adreßerzeugung. Allerdings werden die Adressen von der Instruktion nur dazu benutzt, Daten oder Programmteile zu erhalten; die Adresse selbst ist für den Programmierer nicht verfügbar, in vielen Fällen wird sie sogar nach dem Abarbeiten des Befehls „vergessen". Manchmal ist es allerdings gerade die Adresse, die man für das Programm benötigt. Daher hat der MC68000 zwei Befehle, die dazu dienen, die Adresse zu erhalten, ohne daß sie zum Erlangen von Daten gebraucht wird. Wenn man den 68000 die Adresse berechnen läßt, wird dieser Vorgang wesentlich schneller ausgeführt.

Zwei Befehle, LEA (Load effective Address) und PEA (Push effective Address), berechnen und plazieren die Adresse jeweils im Adreßregister (LEA) bzw. im Stack (PEA). Man kann die effektive Adresse mit Hilfe jeder verfügbaren Adressierart und in Zusammenhang mit jedem Register berechnen. Die Befehle LEA und PEA sind besonders hilfreich, wenn man positionsunabhängige Code benutzt. Manchmal läßt sich auf diese Weise eine Adressierart vorteilhaft nutzen, die wesentlich schneller als beispielsweise die programmzähler-relative Adressierart läuft. Dann kann man die Adresse mit Hilfe von LEA erreichen und den Bereich des Speichers durch indirekte Adressierung über das Adreßregister, in dem sich der Wert, der durch LEA berechnet wurde, befindet, adressieren. PEA und LEA sind auch sehr hilfreich zur Weitergabe von Pointern oder Daten in andere Routinen oder zum Ablegen im Speicher. Manchmal ist es hilfreich, zu verifizieren, daß eine effektive Adresse auch korrekt ist oder zumindest im zulässigen Bereich liegt. Ohne diese beiden Befehle wäre es außerordentlich schwierig, Adressen, die vom Prozessor erzeugt sind, zu benutzen.

Befehle für gemeinsame Ressourcen

Systeme, die über mehr als einen Mikroprozessor verfügen, benutzen sehr häufig gemeinsame Ressourcen, z. B. Speicher, Puffer, E/A-Einheiten, Tasks usw. In diesem Fall muß ein Programm sicher unterscheiden können, welcher Prozessor gewisse Rechte in bestimmten Teilen im Speicher hat, oder im Puffer, in E/A-Einheiten bzw. einer Task. Der MC68000 hat einen Befehl TAS (Test and Set), der eine solche Zuweisung von Ressourcen zwischen Multiprozessoren einfach und sicher ausführen kann.

Wichtig bei dieser Instruktion ist, daß sie unteilbar ist, d. h. sie kann alle Zugriffe im vorgegebenen Adreßbereich ausschließen, bis die Arbeit in diesem Bereich beendet wird. Die Test-and-Set-Instruktion prüft ein gegebenes Byte, setzt die Status-Register-Bits N und Z entsprechend und setzt dann das höchstwertige Bit im Byte auf 1.

In den meisten Fällen benutzen die Mikroprozessoren die TAS-Instruktion folgendermaßen: Zunächst wird ein Byte gewählt, das den Status der gemeinsam genutzten Ressource repräsentiert (dieses Byte wird oft auch „Semaphor" genannt). Wenn der TAS-Befehl darauf hinweist, daß das Byte negativ ist (wenn das höchstwertige Bit 1 ist), weiß der anfragende Mikroprozessor, daß die Ressource in Benutzung ist. Der Prozessor kann dann entweder das Semaphor-Byte erneut testen, bis die Ressource verfügbar ist oder kann auf eine andere Task übergehen. Wenn das TAS-Byte positiv ist (höchstwertiges Bit ist 0), dann weiß der Mikroprozessor, daß die Ressource frei ist. Weil der TAS-Befehl direkt das höchstwertige Bit auf 1 setzt, können alle Mikroprozessoren, die Zugriff auf das Semaphor-Byte haben, die richtige Information über die gemeinsame Ressource erhalten. Der Mikroprozessor, der gerade auf die Ressource zugreift, ist dafür verantwortlich, das höchstwertige Bit zurückzusetzen, wenn er fertig ist.

Der einzige Grund dafür, daß dieser Prozeß effektiv arbeitet, ist die Tatsache, daß der unteilbare Read-Modify-Write-Buszyklus, der die TAS-Instruktion ausführt, mit Hardwaresignalen verhindert, daß irgendeine andere Einheit auf das Semaphor-Byte zugreift. Diese bedeutet, daß niemals zwei Prozessoren das Semaphor-Byte lesen können und gleichzeitig die Information bekommen, daß die Ressource verfügbar ist.

Supervisor- und User-Betriebsarten

Der MC68000 führt Befehle auf eine der beiden Betriebs- bzw. Privileg-Ebenen aus. Die obere Ebene mit der Bezeichnung „Supervisor-Ebene" stellt eine geschützte Umgebung, in der das Betriebssystem arbeiten kann, dar, wobei es selbst und seine Ressourcen vom weniger „vertrauenswürdigen" Benutzer-Code isoliert sind. Nach einer Reset-Operation beginnt der MC68000 in der Supervisor-Betriebsart zu laufen, in der das Betriebssystem sowie alle Interrupt-Routinen ebenfalls arbeiten. Die niedrigere Ebene trägt die Bezeichnung „Benutzer-Ebene" und stellt den Bereich dar, in dem die meisten Anwendungsprogramme ablaufen. Aus diesem Grund befindet sich der Prozessor die meiste Zeit auch in diesem Zustand.

Die einzige Möglichkeit, vom Supervisor- in den Benutzer-Zustand zu kommen, ist die Veränderung des S/U-Statusbits (*Tabelle 6*) im Statusregister des Prozessors. Dies nutzt auch das Betriebssystem aus, wenn es

Tabelle 6. Supervisor-Traps

address error	Wort- oder Langwort-Zugriff auf eine geradzahlige Adresse
illegal instruction	ungültiger Befehl
zero divide	Division durch Null
CHK instruction	Operation außerhalb der Grenzen
TRAPV instruction	Überlauf (V-Bit gesetzt)
privilege violation	Versuch der Ausführung eines privilegierten Befehls im Benutzer-Modus
trace	Befehl beendet und T-Status-Bit-Register ist gesetzt
line 1010 emulator	Versuch, einen Opcode auszuführen, der mit „1010" beginnt
line 1111 emulator	Versuch, einen Opcode auszuführen, der mit „1111" beginnt
TRAPn instruction	TRAPn-Befehl ausgeführt (n = 0...15)

ein Programm auf Benutzer-Ebene startet. Sollte ein Interrupt in der Mitte einer Routine auf Benutzer-Ebene bearbeitet werden, läuft diese Interrupt-Routine ebenfalls auf Supervisor-Ebene, mit der Rückkehr in die unterbrochene Routine allerdings schaltet der MC68000 wieder auf Benutzer-Ebene.

Programme auf Benutzerebene können zum Betriebssystem nur über einen der 16 TRAP-Instruktionen gelangen. Man kann diese Befehle als Supervisor-Aufrufe auffassen, sie übergeben nämlich die Steuerung direkt an eine bestimmte Routine. Mit dem Anschluß einer TRAP-Routine kehrt der Prozessor in der Regel zur Original-Routine auf Benutzer-Ebene zurück, die dann fortgeführt wird. Es gibt 16 verschiedene Supervisor-TRAP-Befehle, die neben anderen Arten der TRAP-Befehle in Tabelle 6 zu finden sind.

Andere Methoden, um zur Supervisor-Ebene zu kommen, existieren, allerdings sind sie entweder bedingt (wie z. B. Fehler-TRAPs) oder asynchron (wie Interrupts). Trotzdem werden alle TRAPs vom Supervisor in ähnlicher Weise behandelt. Zunächst veranlaßt eine TRAP den Prozessor, den Inhalt des Programmzählers und des Status-Registers im Supervisor-Stack zu sichern. Danach geht er zur externen Vektortabelle und holt einen Wert, der den Grund für die TRAP identifiziert, der dann in den Programmzähler geladen wird. Damit kann jede Art TRAP eine separate Bearbeitungsroutine bieten, die dazu dient, das Problem zu lösen, die die TRAP verursachte, und zum Originalprogramm zurückzukehren.

Einige dieser TRAP-Formen sind an bestimmte Bedingungen gebunden. Abhängig davon, ob das Überlauf-Bedingungsbit gesetzt ist, veranlaßt die Instruktion TRAPV entweder nichts oder verursacht einen TRAP, das den MC68000 in den Supervisor-Zustand umschaltet. Dies ermöglicht dem Programm, alle Überlauf-Bedingungen in gleicher Weise mit einer einzigen Routine auf Systemebene zu bearbeiten. Eine andere derartige Instruktion ist die Check-Instruktion (CHK), die verifiziert, ob der Inhalt eines Registers größer als 0 und kleiner als ein spezifizierter Wert ist. Wenn dies innerhalb der Grenzen zutrifft, passiert nichts, und der nächste Befehl wird abgearbeitet. Wenn der Wert außerhalb der Grenzen liegt, springt das Programm direkt über eine Vektortabelle zu einer bestimmten TRAP-Routine zur Bearbeitung. Dies gibt dem Programmierer eine einfache Möglichkeit zu prüfen, ob ein Array-Index innerhalb der richtigen Grenzen liegt. Darüber hinaus werden Versuche, durch 0 zu dividieren und auf nicht richtig normierte Daten zugreifen zu wollen durch eine TRAP-Routine beantwortet.

Behandlung illegaler und nicht implementierter Opcodes

Um für zukünftige Erweiterungen der MC68000 genügend Raum zur Verfügung zu haben, benutzten die Entwickler dieses Prozessors nicht alle Bitmuster, die für die 16-Bit-Opcodes zur Verfügung stehen.

Tabelle 7. Privilegierte Instruktionen im 68000

STOP
RESET
RTE
MOVE (wenn ein Wort zum Statusregister geschoben wird)
MOVE USP
AND, EOR, or OR (wenn ein Immediate-Wert mit dem Statusregister kombiniert wird)

Bei anderen Mikroprozessoren kann man beobachten, daß undefinierte Opcodes ausgeführt werden, häufig mit katastrophalen Ergebnissen, wobei die Kontrolle über den Prozessor verloren geht oder wertvolle Arbeit zerstört wird. Um sicherzustellen, daß ein System vollständig narrensicher ist, wenn undefinierte Opcodes auftreten, akzeptiert der MC68000 keine illegalen Befehle, sondern führt statt dessen eine spezielle Trap-Routine zur Korrektur aus.

Um es zu erleichtern, ganze Blöcke neuer Instruktionen für MC68000-Prozessoren hinzufügen zu können, sind zwei Untergruppen möglicher Opcodes noch nicht implementiert. Jeder 16-Bit-Opcode, der mit dem Binärwert 1010 oder 1111 beginnt, wurde beim 68000 nicht definiert. Versuche, Opcodes mit diesen Vorziffern auszuführen, werden separat erkannt. Dabei führt der Prozessor entweder eine „Line-1010-Emulator“- oder eine „Line-1111-Emulator“-Trap-Routine aus, durch die der Programmierer in der Lage ist, Softwarefunktionen zu emulieren, die auf dem Prozessorchip nicht vorhanden sind. In der Regel werden die 1111-Opcodes als Fließkommabefehle definiert; die 1010-Opcodes sind für die Benutzung in Zusammenhang mit Prozessoren gedacht, die nach dem MC68020 kommen.

Privilegierte Befehle

Privilegierte Befehle haben eine spezielle Eigenschaft: Sie können lediglich dann ausgeführt werden, wenn der Prozessor auf Supervisor-Ebene arbeitet. Versuche, diese auf Benutzerebene ablaufen zu lassen, verursachen sogenannte „Privilege-Violation-Traps“, durch die der Supervisor bestimmte Aktionen auslösen kann.

Die privilegierten Befehle sind in *Tabelle 7* zusammengefaßt. Diese Befehle sind deshalb auf nur eine Ebene beschränkt, weil sie Ressourcen oder Services modifizieren bzw. steuern, die nur vom Betriebssystem beeinflußt werden dürfen. Viele dieser Befehle modifizieren den unteren Teil des Statusregisters (SR), in dem sich das S/U-Supervisor-Bit befindet, die Interrupt-Maske und ein Trace-Mode-Umschalter. Diese Ressourcen dürfen nicht in die Hände des Benutzers fallen, sondern werden vom Supervisor gesteuert.

Eine weitere privilegierte Ressource ist der „Supervisor-Stack-Pointer“ (SSP). Dieser Pointer ist nur dann sichtbar (als Adreßregister A7), wenn der MC68000 auf Supervisor-Ebene arbeitet, ähnlich wie der Benutzer-Stack-Pointer (USP), der nur dann sichtbar ist (als Regi-

ster A7), wenn der MC68000 auf der Benutzer-Ebene arbeitet. Allerdings muß das Betriebssystem auch auf einen verborgenen USP zugreifen können, wenn eine neue Task auf Benutzer-Ebene initialisiert werden soll. Dies erfolgt mit Hilfe der speziellen privilegierten Instruktion MOVE USP.

Der Befehl Stop hält den Prozessor bei der Ausführung weiterer Befehle an, während er auf einen Interrupt, eine Trace-Exception oder einen Reset zur Initialisierung einer neuen Aktivität wartet. Der Befehl lädt auch das Statusregister mit einem 16-Bit-Immediate-Wert, durch den der Programmierer in der Lage ist, gewisse Interrupts freizugeben, bevor der Mikroprozessor gestoppt wird. Nur der Supervisor kann diese Betriebsart initialisieren, weil sie in den Händen des Benutzers die zeitliche Integrität des Betriebssystems stören und einen Absturz hervorrufen könnte. Der Befehl muß auch deswegen auf Supervisor-Ebene beschränkt werden, weil er das gesamte Statusregister betrifft.

Der Reset-Befehl ist eine leistungsfähige Operation. Wenn sie ausgeführt wird, erscheint auf der Reset-Leitung des MC68000 ein Impuls, ohne daß der Prozessor selbst zurückgesetzt ist. Man benutzt diesen Befehl typischerweise nach einem „katastrophalen" Fehler, bei dem das Betriebssystem selbst versucht, den normalen Betrieb wieder aufzunehmen. Hiermit ist es möglich, daß das Betriebssystem seine Umgebung initialisiert (d. h. Reset des gesamten Systems außer dem Mikroprozessor), ohne daß der Prozessor selbst zu einem Neustart gezwungen ist. Auch dieser Befehl ist ganz offensichtlich ungeeignet für die Benutzer-Ebene. Die letzte privilegierte Instruktion ist RTE (Return from Exception). Eine „Exception" ist ein Vorgang, der den Mikroprozessor dazu veranlaßt, eine andere Operation als die nächste normale Instruktion auszuführen. Interrupts und Traps sind Beispiele für Exceptions. Der RTE ist ein ähnlicher Vorgang wie ein „Return from Interrupt", der bei den meisten Mikroprozessoren vorgesehen ist. Die Operation lädt sowohl den Programmzähler als auch das Statusregister mit den Werten aus dem oberen Ende des Stack. Weil alle Exceptions den Prozessor dazu veranlassen, im Supervisor-Betrieb zu arbeiten, wird die RTE-Instruktion nur in diesem Modus ausgeführt. Daher handelt es sich um eine privilegierte Instruktion.

Es ist wichtig, zu erkennen, daß privilegierte Befehle nur auf Supervisor-Ebene ausgeführt werden können, auf der üblicherweise das Betriebssystem residiert. Die beiden Ebenen der Privilegierung und die beschränkte Verwendungsfähigkeit privilegierter Befehle ermöglichen den Aufbau von Systemen, bei denen es nicht möglich ist, daß Anwendungsprogramme auf Benutzerebene Zerstörungen auf der Betriebssystemebene anrichten können.

Zusammenfassung

Die Architektur des MC68000 ist, wie dieser Beitrag gezeigt hat, insbesondere im Hinblick auf den Programmierer entwickelt worden. Verzweigungs- und Sprung-Befehle dieses Prozessors geben eine vollständige Kontrolle über den Programmfluß und vereinfachen häufig benutzte Schleifen- und Ketten-Verschiebungs-Konstrukte. Die Befehle LINK und UNLINK machen den Aufbau modularer Programme mit lokalen Variablen einfach. Andere Befehle führen komplexe Adreßberechnungen schnell aus, helfen bei der Nutzung gemeinsamer Ressourcen, schützen die Datenintegrität des Betriebssystems und ermöglichen die „Wiederbelebung" bei Fehlern. Darüber hinaus ist die Architektur sowie der Zeichensatz so konzipiert, daß Erweiterungen mit leistungsfähigeren und anspruchsvolleren Funktionen ohne weiteres möglich sind. (Übersetzung: Be)

Neuerscheinung

in der Reihe Franzis Ingenieur-Elektronik

Handbuch der digitalen Elektronik

Von Jan Hendrik Jansen

Das Handbuch der digitalen Elektronik ist ein großes und eindeutig praxisbezogenes Lehr- und Nachschlagewerk. – Umfassend und kompetent wird dem Techniker hier Schritt für Schritt das notwendige Ingenieurwissen der digitalen Elektronik vermittelt. Das geschieht ausführlich, anschaulich sowie einprägsam und gut verständlich. Das Handbuch der digitalen Elektronik gibt den aktuellen Stand der Technik wieder. – Der Anwender dieses Standardwerkes lernt die Entwicklung digitaler Schaltungen und Systeme mit allen Problemen und Variationen kennen. Das sichert ihm langfristig den Berufserfolg bei der Entwicklungsarbeit. Selbstverständlich kann mit jedem Band, dem Bedarf entsprechend, auch separat gearbeitet werden.



Teil 1: Bausteine in TTL und CMOS

Die Kombination elementarer elektronischer Grundfunktionen zu digitalen Bausteinen. – Größe: 23 × 16,5 cm. 310 Seiten, 293 Abbildungen. Lwstr-kartoniert mit Schutzumschlag.

DM 65.–

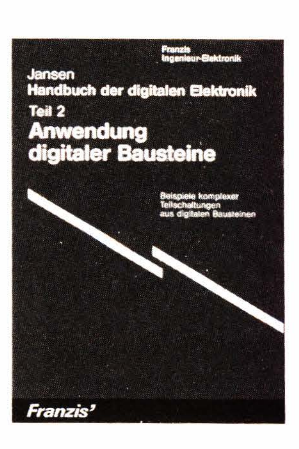
ISBN 3-7723-7921-4

Teil 2: Anwendung digitaler Bausteine

Beispiele komplexer Teilschaltungen aus digitalen Bausteinen. Größe: 23 × 16,5 cm. 333 Seiten, 258 Abbildungen. Lwstr-gebunden mit Schutzumschlag.

DM 65.–

ISBN 3-7723-7931-1



In diesem ersten Band wird der Ingenieur mit den entscheidenden Grundlagen der Digitaltechnik vertraut gemacht. Mit Hilfe fundamentaler TTL- und CMOS-Bausteine werden digitale Teilschaltungen entworfen und bereits so verdrahtet, daß schon erste Anlagen entstehen. Die zahlreichen und einprägsamen Schaltungsbeispiele ebnen dazu den Weg. Diese Beispiele bilden gleichzeitig die Brücke zur Praxis. Der Techniker erkennt jetzt ganz deutlich, daß die Kombination einer Reihe von Grundfunktionen zu immer weiteren und komplexeren Funktionen bzw. Systemen führt.
Die Vermittlung dieses Grundwissens geschieht so ausführlich und gründlich, daß der Leser nach der Lektüre problemlos an weitere Entwicklungsstufen herangeht.

Auszug aus dem Inhaltsverzeichnis:

Einführung in die digitale Schalttechnik und das binäre Rechnen. Speicherfunktionen, Kodieren und Information. Gleitkomma (floating point). Der Begriff „Überlauf" (overflow). – Elementare Verknüpfungsschaltungen. (Die Symbole der NAND- und NOR-Verknüpfung nach Milspec 806B. Der Entwurf eines Logik-Schemas in IEC-Symbolik. Disjunktive Normalform oder Standardsumme. Karnaugh-Veitch-Diagramme für zwei Eingangsvariable.) – Diagramme und Codes. (Darstellung von Information in einem Zeitdiagramm. Der 7-Segment-Code zur Digitalanzeige. Oktal-Code und Hexadezimal-Code.) – Familien logischer Schaltungen. (Dioden-Logik (DL) und Dioden-Transistor-Logik (DTL) mit diskreten Komponenten. Transistor-Transistor-Logik (TTL). Die fundamentale TTL-Schaltung. Low-Power-Schottky-Logik (LS-TTL). Die Familien 74H, 74S und 74L. Neue Logikfamilien in bipolarer Technik – AS, ALS und FAST. CMOS-4000-Serie und 54HC/74HC-Serie. ECL-10000-Serie. Der monostabile und astabile Multivibrator mit NE555.) – Tips zum Entwerfen und Verdrahten von logischen Schaltungen. (Übersprechen zwischen den Signalleitungen. Charakteristische Impedanz und Selbstinduktion von Signalleitungen. Vermeiden von Reflexionen in digitalen Schaltungen durch Verwendung verdrillten Drahtes. CMOS durch einen niederohmigen Steuergenerator steuern. Entstehung unerwünschter Nadelimpulse (glitches). – Sachverzeichnis.

Teilschaltungen komplexerer Art behandelt dieser Band in großer Fülle. Der Techniker wird beim Durcharbeiten und Befragen ohne großen Aufwand mit der Anwendung dieser Circuits vertraut gemacht. Der Aufbau von Schaltungen gipfelt in selbständigen Systemen. Bei der Zusammenschaltung von Systemen kommt auch die Datenübertragung mittels Glasfaser zur Sprache. Einen anderen Schwerpunkt des Bandes bilden die Speicherfunktionen von Flip-Flops. Sowohl einzelne Flip-Flops als auch Zähler, Teiler und Register, die aus ihnen zusammengesetzt sind, werden ausführlich dargestellt und interpretiert. Praktische und für den Nachbau geeignete Schaltungen dazu werden ausführlich besprochen. Der zweite Band des Handbuches bietet auf einen Schlag viele Lösungen für vielfältige Probleme.

Auszug aus dem Inhaltsverzeichnis:

Verdrahtungstechniken und Verbindungssysteme. (Mehrschicht-Platinen und doppelseitig bedruckte Platinen. Steckverbindung zwischen den Rahmengestellen und Verbindungssysteme zur Außenwelt. Glasfaser. Koppelelemente für Glasfasern. Signalverzweigung bei der Glasfaser-Kommunikation.) – Logische Schaltungen und deren Verwendungsbereiche – kombinatorische Logik. (Anzeige logischer Spannungswerte mittels lichtemittierender Dioden (LED). Einzelne und kombinatorische Funktionen mit UND-, ODER-, NAND- und NOR-Schaltungen. Vermehrung von Empfängen. Mustergenerator zur Erzeugung von Zeitsignalen. Kombinatorische Funktionen mit UND-ODER-NICHT-Schaltungen. Exklusive ODER-Schaltung, logischer Addierer und Vergleichsschaltung (Comparator). Schnelle CMOS-74HC-Serie.) – Speicherelemente. (4-bit-Register aus RS-Flipflops mit einseitiger Einlesung. SRT-Flipflop. Untersuchung eines aus NAND-Schaltungen zusammengesetzten JK-Flipflop. D-Flipflop, aus NAND-Schaltungen zusammengestellt. Vierteiler mit flankengetriggerten D-Flipflops. Übersicht über die Steuersymbolik für Speicherelemente gemäß IEC.) – Zähler und Frequenzteiler. (Modulo-10-Zähler oder Zehnteiler. Kaskadenschaltungen binärer und binär-dezimaler Zähler. IEC-Symbolik für Zähler- und Teilerschaltungen. Programmierbare Johnsonzähler zur Frequenzsynthese-Verwendung. Entwurf eines digitalen Uhrwerks mit binären Zählern.) – Register. (Informationsübertragung zwischen Registern untereinander sowie zwischen Registern und anderen Datenquellen. Register mit adressierbaren Flipflops. Register mit Datenfreigabe-Eingang und Three-state-Ausgang. Schieberegister mit seriellen und parallelen Eingängen bzw. Ausgängen.

Teil 3: Komplexe Bausteine und Datenkommunikation

soll im März 1986 erscheinen.

ISBN 3-7723-7941-9

Teil 4: Die Technik der Mikrocomputer

befindet sich in Vorbereitung.

ISBN 3-7723-7951-6



Der große Fachverlag für angewandte Elektronik und Informatik



Franzis-Verlag München

Jetzt wird der VMEbus interessant



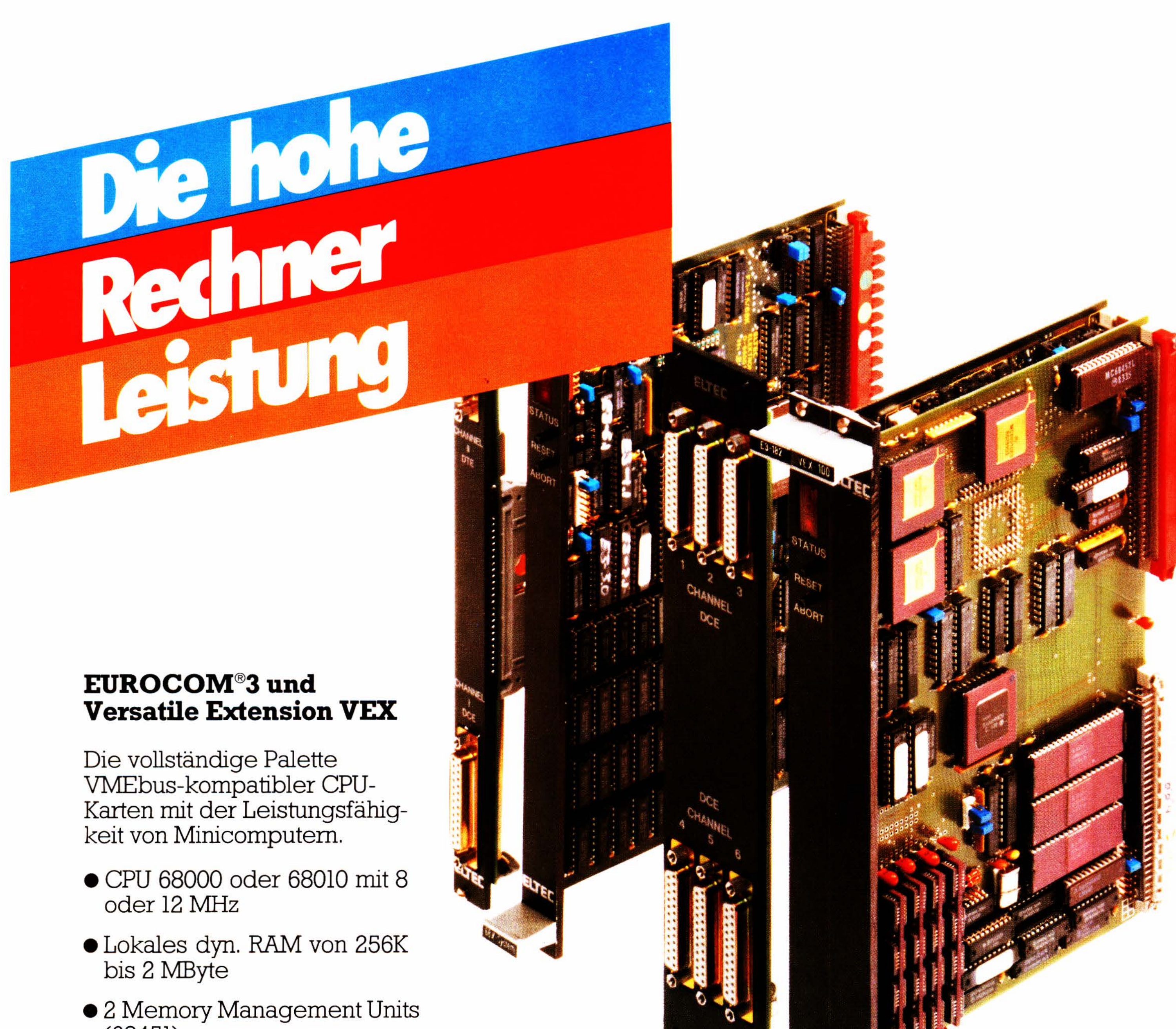


Abb.: Links EUROCOM®3 und rechts EUROCOM®3 mit VEX in voller Ausbaustufe

EUROCOM®3 und Versatile Extension VEX

Die vollständige Palette VMEbus-kompatibler CPU-Karten mit der Leistungsfähigkeit von Minicomputern.

- CPU 68000 oder 68010 mit 8 oder 12 MHz
- Lokales dyn. RAM von 256K bis 2 MByte
- 2 Memory Management Units (68451)
- Schneller Floating Point Processor (NS 32081)
- 4 Kanal DMA-Controller (68450)
- SCSI (SASI) Interface für Harddisk bis 2 GByte und Tapes
- Floppy-Disk Interface 8/5" DD/DS
- 2 syn./asyn., 6 asyn. serielle Schnittstellen mit RS232 oder RS422 Interface
- Real Time Clock mit Akku-Pufferung
- VMEbus Interface mit 4 Level Bus Arbitration Module
- Leistungsfähiges Monitorprogramm mit Assembler/Diassembler
- Volle Software-Unterstützung

Betriebssysteme

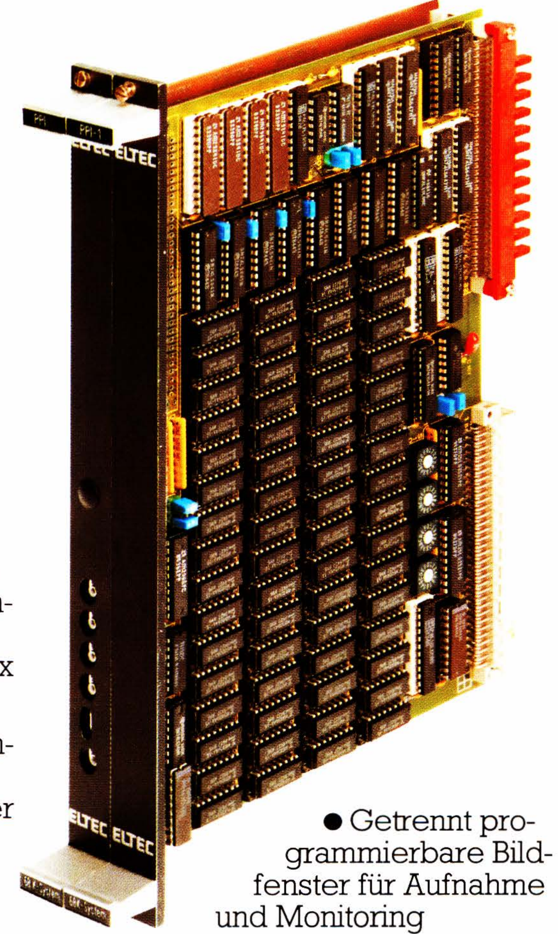
OS9/68000
Modulares Echtzeitbetriebssystem für Mehrbenutzerbetrieb mit Unix-kompatibler Benutzeroberfläche. Unterstützt durch alle modernen Hochsprachen sowie diverse Anwenderprogramme.

CP/M 68K
Flexibles Single-User Betriebssystem mit breiter Softwareunterstützung durch Hochsprachen und Systemutilities. Dateikompatibel zu CP/M 2.2 und CP/M 86. Der optional erhältliche CP/M 2.2 Emulator öffnet dem Anwender zusätzlich das derzeit größte Softwareangebot der Welt.

PDOS
Schnellstes Echtzeitbetriebssystem für den industriellen Einsatz. Unterstützt durch die Hochsprachen Basic, Pascal, Fortran und C.

PEARL
Echtzeit Betriebssystem auf der Basis der Prozessrechnersprache PEARL zur effizienten Programmentwicklung komplexer Echtzeitaufgaben.

Das schnelle Image Processing

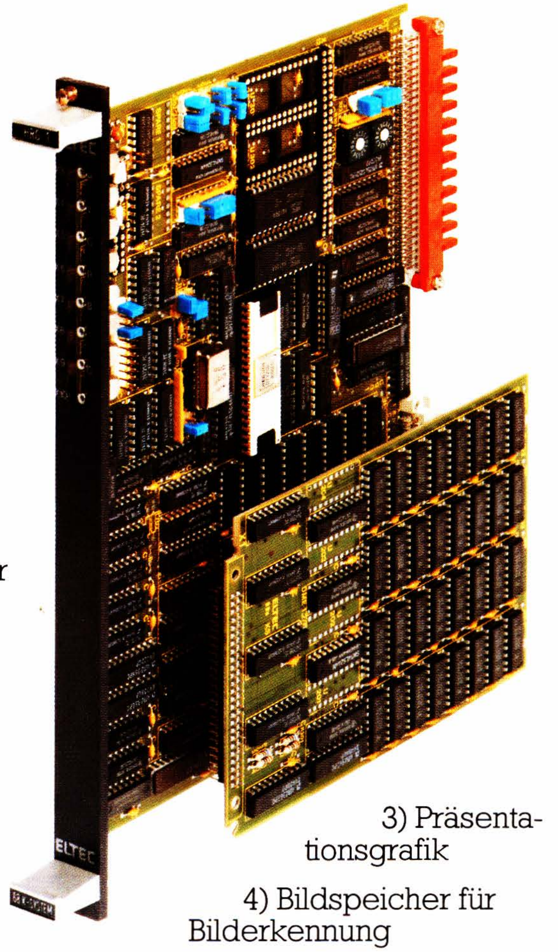


PPI

Platinen-Set zur Echtzeitübernahme und Verarbeitung von Videosignalen.

- Standard Videoeingang für alle CCIR und EIA kompatible Kameras oder Bildquellen
- Auflösung bis zu 512 x 511 Bildpunkten in 255 Graustufen oder Farben
- Verschiedene Ausbaustufen des Bildspeichers ermöglichen das Speichern von max. 16 unabhängigen Bildern der Auflösung 512 x 256 x 8 bit
- Schneller transparenter VMEbus-Zugriff auf den Bildspeicher
- RS 170 kompatibler Video-Ausgang für Kontrollmonitor
- Getrennt programmierbare Bildfenster für Aufnahme und Monitoring





HRG

Hochauflösende Grafik-Platine für Schwarzweiß- und Color-Rastermotive mit Bildspeichererweiterung DME-1.

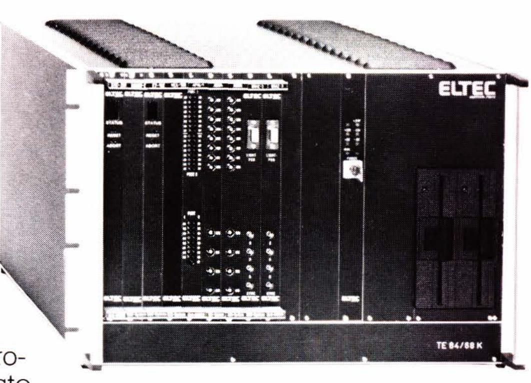
- Auflösung bis zu 2048 x 1024 Bildpunkten frei wählbar
- 8 Graustufen oder Farben und Blinken auf einer Platine
- Schneller Grafik-Controller (NEC 7220 A)
- Direkter Zugriff zum Bildspeicher über VMEbus
- PAN, SCROLL, ZOOM
- Fremdsynchronisierbar
- Softwareunterstützt durch entsprechende Grafiktreiber

Typische Applikations-Beispiele:

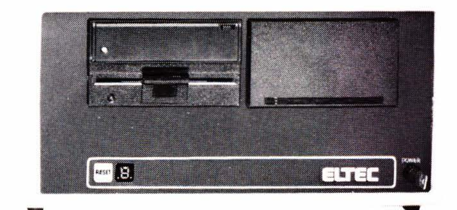
1) Hochauflösende grafische Ausgabe für CAD/CAM
2) Grafische Darstellung von Prozeßdaten
3) Präsentationsgrafik
4) Bildspeicher für Bilderkennung







TE 84
Das modular konfigurierbare 19" Komplettsystem für den industriellen Anwender



UNIFRAME 68K
Die Werkbank des Entwicklers
Die modulare OEM Station

Boards

SAC
Schneller Stand-Alone-Computer für Multiprozessor-Umgebungen oder lokalen Einsatz

GRAZ-3
LOW-Cost Farbgrafikplatine mittlerer Auflösung für CCIR-Monitor

RAM 512
Dynamische Speicherkarte bis 2 MByte, voll dekodiert für echten Longword-Zugriff (32 Bit)

RARO-1
Konfigurierbare Speicherkarte mit 16 Byte-wide-Sockeln nach JEDEC-Standard

APAL
Digitale I/O-Karte mit 48 programmierbaren I/O-Leitungen und zwei 24 Bit-Timern, Statusanzeige über LEDs

IGEN
Interrupt-Generator mit 32 Eingängen, 4 seriellen Schnittstellen und 16 8Bit Timer/Counter

ADDA
Analog I/O-Karte, 8 analoge Eingänge, 4 analoge Ausgänge 12 Bit Auflösung, Wandelzeit $< 9\,\mu$sec

Systeme

- CPU 68000 oder 68010
- Multiprozessorfähig
- Arbeitsspeicher 256 KByte bis 16 Mbyte
- Breites Angebot von verschiedenen Massenspeichermedien für alle Anwendungsbereiche von 2–170 MByte über Standard SCSI Schnittstelle
- Massenspeicherzugriff über DMA möglich
- Optionaler Floating Point Processor NS32081 für schnellste Gleitkommaarithmetik
- Mehrplatzunterstützung bis max. 8 Benutzer über V24/RS232c oder RS422 Schnittstellen
- Je nach Grundausbaustufe 6–19 freie VMEbus Steckplätze für Systemerweiterungen
- Volle 32-Bit Unterstützung des VMEbusses für Erweiterungskarten
- Unterstützt durch alle Standard-Betriebssysteme der Industrie

Software

CP/M 68K
Lieferumfang: C-Compiler, Macro-Assembler, Debugger
Sprachen: BASIC Interpreter, C-BASIC Compiler, MT + Pascal Compiler, Omegasoft Pascal, SVS Fortran 77, Z80 Emulator.
Weitere Sprachen auf Anfrage.
In Vorbereitung: Concurrent CP/M-68K, Modula, LISP, 6809 Emulator

OS9/68000
Realtime Multiuser Multitasking Betriebssystem
Lieferumfang: Bildschirmorientierter Editor, Macro-Assembler, Debugger
Sprachen: C-Compiler, BASIC 09, Omegasoft Pascal, DYNACALC, STYLOGRAPH, Mail Merge
In Vorbereitung: Pascal 09, Fortran Compiler

Eltec ist ein flexibles, leistungsfähiges Mittelstandsunternehmen mit eigener Entwicklung – Eltec bietet Ihnen persönliche, individuelle Betreuung bei der Realisierung Ihrer Aufgabenstellung.
Sprechen Sie mit Eltec!



elektronik mainz

Eltec Elektronik GmbH · Galileo-Galilei-Straße 11 · 6500 Mainz 42
Postfach 65 · Telefon 06131/50031 · Telex 4187273

Anton Nausch

Die M68000-Familie

Fünf extrem leistungsfähige Mikroprozessoren mit einer internen Wortbreite von 32 Bit und externen Wortbreiten von 8, 16 oder 32 Bit umfaßt die M68000-Familie – zusammen mit einer großen Zahl von Peripherie-ICs. Hier ein kurzer Überblick der CPUs und Peripherie-ICs aus der M68000-Familie.

Die Prozessoren der M68000-Familie

Der Ausgangspunkt für die M68000-Familie ist der 16/32-Bit-Mikroprozessor MC68000. Seit 1979 ist er auf dem Markt verfügbar und ist in vielen Anwendungen zu finden. Seine Architektur sowie Merkmale sind die Grundlage für alle Mikroprozessoren der M68000-Familie.

Diese Merkmale umfassen:
- Acht 32-Bit-Datenregister
- Sieben 32-Bit-Adreßregister
- Zwei voneinander unabhängige Stackpointer (User- und Supervisor-Stackpointer)
- Ein 32-Bit-Programmzähler
- Ein 16-Bit-Statusregister
- 56 sehr leistungsfähige Befehle
- 14 universelle Adressierungsarten
- Fünf verschiedene Datentypen
- Vorkehrungen für die Erkennung und Bearbeitung von Ausnahmezuständen
- 16 MByte nichtsegmentierter, linearer Adreßraum
- Adreß- und Datenbus ohne Multiplex-Umschaltung

Der 8/32-Bit-Mikroprozessor MC68008

Der Mikroprozessor MC68008 ist eine Variante des MC68000. Objekt- und Sourcecode des MC68008 sind voll kompatibel zum MC68000. Das bedeutet, daß alle Programme, die für den MC68000 geschrieben wurden, genauso im MC68008 ablaufen. Das gilt aber auch für die umgekehrte Richtung (MC68008-Programme laufen auch auf dem MC68000). Intern ist der MC68008 genauso aufgebaut wie der MC68000, er besitzt also denselben Registersatz usw. wie der MC68000. Der einzige wichtige Unterschied zwischen MC68000 und MC68008 besteht in der Breite des Datenbusses. Während beim MC68000 der Datenbus 16 Bit breit ist, hat er beim MC68008 eine Breite von 8 Bit. Dadurch ist es möglich, die Leistung des MC68000 in ein reines 8-Bit-System zu bringen. Eine weitere Einschränkung, die bei einem 8-Bit-System jedoch nicht ins Gewicht fallen sollte, ist der Adreßbereich des MC68008 von 1 Megabyte.

Die virtuelle Maschine MC68010

Der MC68010 stellt in der M68000-Familie die Fortsetzung des MC68000 in Richtung höherer Leistung dar. Von den Merkmalen, wie Architektur, Registersatz, Businterface usw., besitzt der MC68010 dieselben Daten wie der MC68000. Von der Software her ist der MC68010 voll aufwärtskompatibel in Objekt- und Sourcecode. Kompatibilität von MC68000 und MC68010 ist auch bei der Anschlußkonfiguration gegeben. Beide Prozessoren sind vollkommen anschlußkompatibel. Der MC68010 kann daher sofort in jeder auf dem MC68000 basierenden Hardware verwendet werden. Zusätzlich kann der MC68010 auch mit virtuellem Speicher zusammenarbeiten, und er stellt eine virtuelle Maschine dar.

Der 32-Bit-Mikroprozessor MC68020

Der MC68020 ist der 32-Bit-Typ der 68000-Familie. Er ist voll aufwärts Objekt-Code-kompatibel zu den anderen Mikroprozessoren der 68000-Familie. Durch seine zusätzlichen, sehr leistungsfähigen Adressierungsarten ist er ideal geeignet für Anwendungen mit höheren Programmiersprachen. Sein Adreßbereich von 4 GByte läßt sehr umfangreiche Programme zu. Zusätzlich ist der 68020 wie auch der 68010 eine virtuelle Maschine und kann virtuellen Speicher verwalten. Dazu wird eine „Memory Management Unit" (Speicherverwaltungseinheit) benötigt. Es ist möglich, dafür die bekannte MMU MC68451 zu verwenden. Sinnvoller ist es jedoch, eine MMU für Paging-Zugriff, die unter der Bezeichnung MC68851 ab Ende 1985 verfügbar sein wird, einzuset-

Die M68000-Familie

	Typen-bezeichn.	Bausteintyp	Beschreibung	Hersteller	Geschwindig-keit	Gehäuse
Prozessoren	MC68000	16/32-Bit-MPU	MPU, 16 Bit externe, 32 Bit interne Wortbreite. 17 allgemeine 32-Bit-Register. 16 MByte linearer Adreßbereich	M, H, MK, R, S, T	8, 10, 12,5 MHz	64polig L, P 68polig R ZB, ZC, FN (2Q85)
	MC68008	8/32-Bit-MPU	MPU, 8 Bit externe, 32 Bit interne Wortbreite. 17 allgemeine 32-Bit-Register. 1 MByte linearer Adreßbereich (4 MByte in FN-Gehäuseversion)	M, MK, S, T	8, 10 MHz	48polig L, P 52polig FN (2Q85)
	MC68010	Virtuelle Maschine 16/32-Bit-MPU	MPU, 16 Bit externe, 32 Bit interne Wortbreite. 17 allgemeine 32-Bit-Register. Virtueller Speicher mit 16 MByte linearem Adreßbereich	M, S	8, 10, 12,5 MHz	64polig L, P 68polig R
	M68012	Erweiterte virtuelle Maschine 16/32-Bit-MPU	MPU, 16 Bit externe, 32 Bit interne Wortbreite. 17 allgemeine 32-Bit-Register. Virtueller Speicher mit 2 GByte linearem Adreßbereich	M	8, 10, 12,5 MHz	84polig R
	MC68020	32-Bit-MPU	Echter 32-Bit-Mikroprozessor. 4 GByte linearer Adreßbereich. Coprozessor-Interface. Instruktions-Cache, dynamische Busbreiten-Anpassung, Leistung 2...3 MIPS	M	12,5 MHz 16,67 MHz (1Q85)	114polig R
	MC68881	Fließkomma-Coprozessor (FPCP)	Entspricht IEEE-Spezifikationen für moderne Fließkommaverarbeitung. Einfache, doppelte und erweiterte Genauigkeit	M	12,5 MHz 16,67 MHz	68polig R
Speicherverwaltung	MC68451	Speicher-verwaltungs-Einheit (MMU)	MMU-Baustein für 68000- und 68010-Systeme ohne Demand-Paged-Zugriff	M, MK	8, 10 MHz	64polig L 68polig R
	MC68851	Paged-MMU (PMMU)	32-Bit-Speicherverwaltungs-Einheit für Demand-Paged-Zugriff auf virtuellen Speicher (68020-Systeme)	M (3Q85)	12,5 MHz 16,67 MHz	121polig R
	MC68461	Speicher-verwaltungs-Controller (MMC)	Gate-Array-Implementierung der PMMU-Funktion. Eignet sich für 68020, 68010 und 68012	M (4Q85)	N/A	R oder „Mezz-Board"
DMA-Steuerung	MC68440	Doppel-Controller DMA (DDMA)	Schneller Zwei-Kanal-DMA-Controller. 5 MBit/s Transferfrequenz	M	8, 10, 12,5 MHz (3Q85)	64polig L 68polig R
	MC68450	DMA-Controller (DMAC)	Vier-Kanal-DMA-Controller. Eignet sich für komplexe verkettete Datentransfers	M, H	8, 10 MHz	64polig L 68polig R
	MC68442	DDMA	DDMA mit 32-Bit-Adressen. Unterstützt den 4-GByte-Adreßbereich des MC68020. Anschlußkompatibel zum 68440/68450	M	8, 10, 12,5 MHz (3Q85)	64polig L, P 68polig R
System-Interface	MC68153	Bus-Interrupt-Modul (BIM)	Verarbeitet Interrupts von vier unabhängigen Quellen für sieben MPU-Interruptebenen	M	200 ns Zugriffszeit (16-MHz-Takt)	40polig L, P
	MC68452	Bus-Arbitrations-Modul (BAM)	Regelt den Zugriff auf ein 68000-System mit bis zu acht lokalen Mastern	M	50 ns Arbitrationszeit	28polig L, P
	MC68173	VMEbus-Controller (E-BUSCON)	Steuert Arbitration zwischen VMEbus/lokalem Bus, VMEbus-Requests. Bus-Transceiver-Steuerung	S (2Q85)	N/A	28polig L, P
	MC68173	VMEbus-Controller (S-BUSCON)	Interface für Operationen zwischen Hochgeschwindigkeits-Peripheriebus (VMSbus) und VMEbus	S (2Q85)	N/A	28polig L, P
	MC68174	VMEbus-Arbiter (E-BAM)	„Round-Robin"- und 4-Ebenen-Prioritäts-Arbitration für VMEbus	M (2Q85)	N/A	20polig L, P

Hersteller: M = Motorola, H = Hitachi, MK = United Technologies/Mostek, R = Rockwell, S = Philips/Signetics, T = Thomson-CSF

Gehäuse: L = Keramik DIP, P = Plastik DIP, R = Pin-Grid-Array, ZB = LCC für Fassung, ZC = LCC für Oberflächenmontage, FN = Plastik-Quad-Pack

	Typen-bezeichn.	Bausteintyp	Beschreibung	Hersteller	Geschwindig-keit	Gehäuse
Datenkommunikation	68561	Multi-Protokoll-Kommunikations-Controller (MPCC-2)	Einkanal-Interface für 68000-Asynchron, Bisynchron, SDLC	R (1Q85)	4 MBs	48polig L
	68562	Serieller universeller Doppel-Komm.-Contr.	Zwei-Kanal-Kommunikations-Controller, asynchrone Bytesteuerung (bisynchron DDCMP, X.25), DMA-Interface, Zähler/Zeitgeber	S (1Q85)	4 MBs	48polig L
	68564	Serielle E/A (SI/O)	Zweikanal-Typ (asynchron, bisynchron, SDLC)	MK	1 MBs	48polig L, P
	MC68652 MC2652	Multi-Protokoll-Komm.-Contr. (MPCC)	Ein-Kanal-Typ (Bytesteuerung, bitorientiert), CRC-Schaltung (der Typ MC2652 hat ein eigenes Businterface)	M, S	2 MBs	40polig L, P
	MC68653 MC2653	Polynomial-Generator-Checker (PGC)	Fehlerkorrektur, Codeerzeugungs-/Komparatorschaltung; paßt zu MPCC 68652 und EPCI68661 (der Typ MC2653 hat eigenes Businterface)	M, S	4 MBs	16polig L, P
	MC68661 MC2661	Erweitertes Peripherie-Komm.-Interface (EPCI)	Universeller asynchroner/synchroner Sender/Empfänger mit Doppel-Puffer; interner Taktgenerator (der Typ 2661 hat eigenes Businterface)	M, S	1 MBs	28polig L, P
	MC68681 MC2681 MC2682	Doppel-UART (DUART)	Zwei Kanäle, vier gepufferte Empfänger, doppelter gepufferter Sender, unabhängige Baudratenwahl 1 MBit/s (2681 hat eigenes Businterface, 2682 bietet Teilfunktion in einem kleineren Gehäuse)	M, S	1 MBs	40polig L, P (MC2682 28polig)
LAN-Controller	68590	LAN-Controller für Ethernet (LANCE)	Vollständiges Ethernet-Interface nach IEEE-Spezifikationen für 68000-Systeme, mit DMA-Controller	MK (2Q85) M (1Q85)	10 MBs	48polig L
	68802	IEEE 802.3 LAN Controller (LAN)	Kommunikationssteuerung zwischen 68000-Systemen und LAN-Protokoll (802.3)	R	10 MBs	40polig P
Plattensteuerung	MC68454	Intelligenter Multi-Disk-Controller (IMDC)	Steuert bis zu vier Plattenlaufwerke, jede Kombination von einfacher/doppelter Schreibdichte, Hard- und Floppy-Disk möglich, 256-Byte-FIFO, 4-GByte-DMA-Controller	M, S	8, 10 MHz	48polig L, P
	68459	Platten-PLL (DPLL)	Zusatzbaustein für IMDC 68454, dient zum Anschluß von mehr als einem Laufwerk an den IMDC	S	N/A	24polig L, P
	68465	Floppy-Disk-Controller (FDC)	Zum Anschluß an zwei Laufwerke für Disketten doppelter Dichte am 68000-Bus	R	N/A	48polig L, P
Allgem. E/H	MC68120/ 68121	Intelligenter Peripherie-Controller (IPC)	Peripherie-Steuerung für M68000- und M6800-Systeme	M, S	1 MHz 1,25 MHz	48polig L
	MC68230	Parallel-Interface/Timer (PI/T)	Unidirektionales/bidirektionales 8-/16-Bit-Parallel-Interface mit doppeltem Puffer, 24-Bit-Zeitgeber mit 5-Bit-Vorteiler, 68000-Interface	M, MK S,T	8, 10 MHz	48polig L, P
	MC68901	Multi-Funktions-Peripherieeinheit (MFP)	Ein-Kanal-USART, acht Source-Interrupt-Controller, acht parallele E/A-Leitungen, vier 8-Bit-Timer	M, MK	4 MHz 1 MBs USART	48polig L, P
Grafik	MC68486 MC68487	Raster-Memory-System (RMS)	Eignet sich für Bit-Mapped- oder objektorientierte Grafiksysteme. Die Merkmale umfassen Objektdefinition und -manipulation, Kollisionserkennung, Lichtgriffeleingang, x/y-Erfassung und -Interrupt, 32 aus 4096 Farben, sichtbare/virtuelle Bildschirme	M(2Q85)	N/A	28polig L, P (beide)

zen. Als Zwischenlösung existiert mit dem MC68461 eine Ersatzschaltung für die MC68851. Die Leistungsfähigkeit des MC68020 basiert nicht nur auf seiner 32-Bit-Architektur, sondern auch auf seinem Befehlscachet, der auf dem Chip vorhanden ist.

Für die M68000-Familie ist in der Zwischenzeit eine große Anzahl verschiedener Peripheriebausteine verfügbar. Der Bus des MC68000 ist asynchron. Das bedeutet, daß der Prozessor die Adresse auf dem Bus mit einem Strobesignal gültig macht und daraufhin die Peripherie mit einem „Data Transfer Acknowledge"-Signal (DTACK) antwortet, wenn die Lese- oder Schreiboperation durchgeführt werden kann. Das ist eine sehr effiziente Methode und sie erleichtert den Systementwurf, da sich sowohl langsame als auch schnelle Peripherie ohne Aufwand anschließen läßt. Verschiedene Karten in einem System können unterschiedliche Antwortzeiten besitzen, und trotzdem läuft das System immer mit der größtmöglichen Geschwindigkeit. Neue Peripheriesteuerbausteine werden nicht nur die asynchrone Busstruktur des 68000 unterstützen, sondern auch andere Eigenschaften, wie zum Beispiel die „Function-Code-Bits". Diese Peripheriebausteine wurden so entworfen, daß sie in Systemen mit dem 8-Bit-Typ 68008, den 16-Bit-Typen 68000 und 68010 oder dem 32-Bit-Typ 68020 eingesetzt werden können. Das bedeutet für den Anwender, daß er wegen der großen Stückzahl billiger einkaufen kann und die Entwicklungskosten niedriger sind.

Der intelligente Peripheriesteuerbaustein MC68120/MC68121/MC687120

Diese Bausteinfamilie basiert auf dem MC6801, einem Single-Chip-Mikroprozessor. Sie enthalten alle drei je eine serielle Schnittstelle, viele E/A-Leitungen, einen Timer, einen Dual-Port-RAM-Puffer mit 128 Byte und zwei voneinander unabhängige Busse. Der eine Bus dient zum Zugriff des MC68000 auf das Dual-Port-RAM des MC68120/121/7120 und der zweite Bus dient zum Anschluß von lokaler Peripherie an den MC68120/121/7120. Der MC68120 enthält zusätzlich dazu ein ROM mit 2048 Byte auf dem Chip, beim MC68121 fehlt dieses ROM. Der MC687120 verfügt statt des 2048-Byte-ROMs über ein 2048-Byte-EPROM. Der hauptsächliche Anwendungsbereich für diese Peripheriecontroller ist die Ansteuerung von Terminals, Druckern usw. in einem leistungsfähigen MC68000-System, um den Zentralprozessor von Ein-/Ausgabeaufgaben zu entlasten.

Der Interrupt-Controller MC68153

Für den Aufbau eines Interruptsystems mit dem MC68000 benötigt man Bausteine, die an den MC68000 einen Interruptvektor übergeben können. Da nicht alle Bausteine diese Aufgabe erfüllen, ist der MC68153 dafür vorgesehen, die Eigenschaften zu nutzen.

Das Parallelport mit Timer MC68230

Dieser Baustein enthält drei 8-Bit-E/A-Ports, vier Handshake-Leitungen und zwei 24-Bit-Timer mit einem 5-Bit-Vorteiler. Die E/A-Ports können uni- oder bidirektional konfiguriert werden und entweder als drei 8-Bit-E/A-Ports oder als ein 16-Bit- und ein 8-Bit-Port fungieren.

Die DMA-Controller MC68440/MC68450

Um Daten in einem System schnell übertragen zu können, benötigt man einen DMA-Controller. Hierzu gibt es die beiden Bausteine MC68440 und MC68450. Der wichtigste Unterschied zwischen diesen beiden Bausteinen besteht darin, daß der MC68440 ein Zweikanal-DMA-Controller ist und der MC68450 ein Vierkanal-DMA-Controller.

Die Speicherverwaltungseinheit MC68451 (MMU)

In einem System mit einem MC68000 hat eine MMU verschiedene Aufgaben zu erfüllen: erstens die Umsetzung von logischen Adressen in physikalische und zweitens den Aufbau von Schutzmechanismen für verschiedene Speicherbereiche. Die MC68451 ist eine MMU, die mit Segmenten arbeitet. Es sind auf der MMU Vorkehrungen für die Verwaltung von 32 Segmenten getroffen. Der Baustein kann aber mit „Paging" verwendet werden, wobei ein Segment einer Page entspricht.

Das Bus-Arbitrations-Modul MC68452

Um den Einsatz des MC68000 in einem Multiprozessorsystem zu erleichtern, wurde der Baustein MC68452 geschaffen. Er bildet abhängig von der Priorität der Busanfragen eine Warteschlange und übergibt den Bus dem Anfrager mit der höchsten Priorität. Damit ist ein Zusammenarbeiten von mehreren CPUs in einem System ohne Buskonflikte gewährleistet.

Die Plattenspeicher-Steuereinheit MC 68454/ MC68459

Diese beiden Bausteine bilden einen komplexen Platten-Controller für ein MC68000-System. Der MC68459 ist dabei die zugehörige Disk-PLL und der MC68454 der Steuerbaustein.

Die Speicher-Verwaltungseinheit MC68461

Mit dem MC68461 steht dem Entwickler von 68020-Systemen eine leistungsfähige Speicherverwaltungseinheit zur Verfügung. Bei diesem Baustein handelt es sich um eine Gate-Array-Nachbildung der MMU MC68851.

Der Funktionsumfang des MC68461 ist im Vergleich zum MC68851 eingeschränkt. Es handelt sich jedoch um eine Speicher-Verwaltungseinheit für Page-Zugriff, die einen linearen Adreßbereich von 4 GByte zuläßt. Die Pages in der MMU MC68461 sind 1 KByte groß, und der Baustein unterstützt den Anschluß eines externen Cache-Speichers für die Adreß-Translation. Außerdem bietet dieser Baustein die Möglichkeit zum Anschluß an den 68010.

Dipl.-Ing. Anton Nausch, geboren in Zwiesel (Niederbayern), studierte nach dem Abitur Elektrotechnik und Datenverarbeitung an der Technischen Universität München. Während seiner Beschäftigung bei der Fa. Siemens lernte er das Arbeiten mit 16-Bit-Mikroprozessoren von Grund auf.
Seit 1981 ist er als Applikationsingenieur für 16/32-Bit-Prozessoren und die zugehörigen Produkte bei der Fa. Motorola in München tätig. Er veröffentlichte mehrere Artikel und, zusammen mit einem Kollegen, ein zweibändiges Fachbuch zum M68000.
Hobbys: Auto, Skifahren, Schwimmen, Faulenzen



Das „Raster-Memory-System" MC68486/487

Diese beiden Bausteine stellen einen Grafik-Controller-Chipsatz dar. Es handelt sich um sehr leistungsfähige Bausteine für Anwendungen in verschiedensten Systemen. Die Bausteine lassen eine Grafikauflösung von 640 × 500 Bildpunkten zu. Natürlich ist eine Darstellung in Farbe jederzeit möglich. Es lassen sich dabei aus 4096 verschiedenen Farben jeweils 32 darstellen. Auf dem Bildschirm ist ein verschiebbares Fenster innerhalb eines virtuellen Bildschirms sichtbar. Scrolling ist in horizontaler und vertikaler Richtung Bildpunkt für Bildpunkt möglich. Die Bausteine können Speicher bis zu 16 MByte verwalten und erledigen nebenher auch den Refresh von dynamischen Speichern in der Größe von 16 K × 1 bis 256 K × 1. Es ist ohne weiteres möglich, einen Lichtgriffel anzuschließen.

Das Raster-Memory-System läßt die Definition von Objekten zu. Ein Obekt ist ein beliebig definiertes Bitmuster, das von den Grafik-Controllern selbst bewegt oder vergrößert (Zoom) werden kann. Zusätzlich ist für diese Objekte eine Kollisionserkennung möglich. Der Chipsatz besteht aus dem RMC MC68486 (Raster-Memory-Controller) und dem RMI MC68487 (Raster-Memory-Interface). Beide Bausteine sind in einem 48poligen Gehäuse untergebracht. Der RMI ist ein TTL-Baustein, und er steuert den Refresh des dynamischen RAMs, bedient das Interface zum Mikroprozessor und erzeugt den Takt. Der RMC ist ein HCMOS-Baustein, und er enthält die gesamte Logik für die Darstellung des Bildes auf dem Bildschirm.

Die Bausteine für serielle Übertragung MC68562/MC68652/MC68653/MC68661/ MC68681

Diese Bausteine finden ihre Anwendung in ziemlich jeder Applikation, die mit serieller Übertragung zu tun hat. Die Eigenschaften der einzelnen Bausteine sind im Überblick in der Tabelle zu finden.

Der Ethernet-Controller MC68590

Für die Unterstützung von Ethernet dient der MC68590. Dieser LAN-Controller für Ethernet enthält einen integrierten DMA-Controller, der den vollen Adreßbereich des 68000 von 24 Bit unterstützt, um große Blöcke von Daten zu erfassen. In diesem Baustein sind die Funktionen der Ethernet-Spezifikation implementiert. Er behandelt zuverlässig Fehler und führt eventuell nötige Wiederholungen selbst durch.

Die Speicher-Verwaltungseinheit MC68851

Der Baustein MC68851 ist die ideale Speicher-Verwaltungseinheit für den Anschluß an den Prozessor MC68020. Der Adreßbereich dieser MMU umfaßt 4 GByte. Die Größe der verwalteten Pages in dieser MMU kann variiert werden von 256 Byte...32 KByte. Auf dem Chip befindet sich ein assoziativer Translation-Cache für 64 Einträge. Da der MC68851 ein Coprozessor-Interface besitzt, ist er für den direkten Anschluß an den MC68020 geeignet. Die MMU MC68851 stellt damit eine Erweiterung des Befehlssatzes des MC68020 dar. Auf dem MC68851 sind Register für das Software-Debugging vorhanden. Dadurch ist es für ein System mit diesen beiden Bausteinen sehr leicht, Breakpoints zu verwalten. Die MMU MC68851 wird wie der MC68020 in HCMOS-Technologie hergestellt.

Der Fließkomma-Coprozessor MC68881

Um die Verarbeitung von arithmetischen Ausdrücken zu beschleunigen, wird der Baustein MC68881 entwikkelt. Dieser Coprozessor wird ein direktes Interface zum MC68020 besitzen und kann in MC68000-, MC68008- und MC68010-Systemen als Floating-Point-Processor eingesetzt werden. Seine wichtigsten Merkmale sind: acht 80-Bit-Datenregister, drei 32-Bit-Statusregister und ein umfangreicher Befehlssatz. Die Verarbeitung entspricht voll dem IEEE-Standard (Draft 10.0).

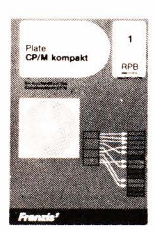
Der Mehrfunktions-Peripheriebaustein MC68901

Dieser Baustein enthält mehrere interessante Eigenschaften. Es existieren auf diesem Baustein zwei Multimode-Timer, zwei Delay-Timer, ein Interruptcontroller, acht programmierbare E/A-Leitungen und ein Einkanal-USART.

Wie man sieht, existiert für die M68000-Mikroprozessorfamilie eine große Anzahl von Peripheriebausteinen, so daß man ein komplettes System aufbauen kann. Durch die Second-Source-Abkommen im Bereich der M68000-Familie werden in Zukunft noch weitere neue Peripheriebausteine hinzukommen.

RPB 87
Methodische Fehlersuche in der Industrie-Elektronik. Fehlerortung durch zielbewußte Systematik und Logik. (Benda)
DM 9.80 ISBN 3-7723-0872-4
Anhand von zwei Beispielen aus der Analog- und Digitaltechnik werden die Grundvoraussetzungen erarbeitet. Darauf wird an vielen Modellen aus allen Bereichen der Industrie-Elektronik geübt.

RPB 149
Kondensatorenkunde für Elektroniker. Eine ausführliche Darstellung der Kondensatoren und ihrer Kennwerte, Bauformen, Eigenschaften, Anwendungsbeispiele und Kennzeichensysteme. (Leucht)
DM 12.80 ISBN 3-7723-1491-0
Sämtliche Bauformen der Kondensatoren mit ihren Kennwerten, ihren speziellen Eigenschaften, den Anwendungsgebieten und Kennzeichensystemen sind beschrieben.



Neu – der ideale Kurzführer für CP/M-Kommandos und -Befehle.

RPB 1
CP/M kompakt. Ein Kurzhandbuch des Betriebssystems CP/M. (Plate)
DM 9.80 ISBN 3-7723-4011-3
Ein ideales CP/M-Handbuch und gleichzeitig eine kurze und bündige Betriebsanleitung. Die Kommandos werden im ersten Teil des Bandes kurz beschrieben und erläutert. Eine übersichtliche Beschreibung der BDOS-Systemaufrufe befindet sich im zweiten Teil. Daran schließt sich eine Übersicht des 8080-Assemblers an.

RPB 183
Modelleisenbahnen digital ferngesteuert. Eine digitale Mehrzugsteuerung und ein lenkbarer Oberleitungsbus ermöglichen vielfältige Betriebsabläufe. (Christoffers)
DM 12.80 ISBN 3-7723-1831-2
Modelleisenbahner und Hobbyelektroniker werden an diesem Band gleichermaßen Freude haben. Ihnen öffnet sich ein weites Betätigungsfeld. Wer noch wenig elektronische Kenntnisse hat, dem werden sie beigebracht.



Werkstattkosten sparen!

RPB 193
Kfz-Motor-Testgerät selbstgebaut. Abgaswerte und Betriebssicherheit mit einem vielseitigen Meßgerät überprüft. (Schlichtmann)
DM 9.80 ISBN 3-7723-1931-9
Der Autor empfiehlt, sich im Selbstbau ein universelles Kfz-Meßgerät selbst herzustellen.
... Und er zeigt auch ganz genau und ausführlich, wie es dann genutzt werden kann und soll. Damit ist das eigene Kraftfahrzeug elektrisch durchgemessen und eventuelle Mängel können kostensparend beseitigt werden.

RPB 33
Elektronische Voltmeter. Grundlagen und Praxis der elektronischen Volt- und Multimeter. (Limann/Pelka)
DM 12.80 ISBN 3-7723-0339-0
Ein umfassender Band über Diodenvoltmeter, Verstärkervoltmeter sowie Digital- und Choppervoltmeter. Die ganze Vielfalt dieser Geräte ist analysiert und dargestellt.

RPB 99
Wie arbeite ich mit dem Elektronenstrahl-Oszilloskop? Eine Fibel der Oszilloskoptechnik nebst einer umfangreichen und universellen Betriebsanleitung für Amateure und Praktiker. (Sutaner/Wißler)
DM 12.80 ISBN 3-7723-0991-7
Da keine theoretischen und mathematischen Kenntnisse vorausgesetzt werden, eignet sich der Band ganz besonders für Praktiker und junge Servicetechniker.

RPB 139
Digitale Steuerung von Modelleisenbahnen. Elektronische Hilfsmittel, um möglichst viele Züge gleichzeitig fahren zu lassen. (Platerink)
DM 12.80 ISBN 3-7723-1392-2
Schritt für Schritt führt der Autor den Modelleisenbahner an sein Ziel – eine digitalgesteuerte Anlage. Einfach und klar verständlich erläutert er die Grundlagen der Digitaltechnik, die der Hobby-Eisenbahner haben muß.

RPB 300
Kfz-Elektronik im Selbstbau. Warn- und Schutzschaltungen und elektrische Zündungen. (Jansen)
DM 12.80 ISBN 3-7723-3004-5
Hier wird eine große Anzahl von Schaltungen angeboten, die leicht selber nachgebaut werden können.

RPB 188
Prozeßrechner-Systemprogramme. Betriebssysteme – Anwenderpakete – Standardhilfsprogramme. (Schorn)
DM 9.80 ISBN 3-7723-1881-9
Der Autor zeigt, wie mit Hilfe von Digitalrechnern verfahrenstechnische Prozesse in der Industrie automatisiert werden können. Die auf dem Markt vorhandenen Betriebssysteme werden beschrieben. Es folgen dann die Software-Anwenderpakete und zum Schluß werden Hilfen für die Programmentwicklung gegeben.

RPB 185
VMOS-Schaltungen. VMOS-Bausteine im Nf-Bereich, in Signalkreisen sowie bei Tongenerator- und Steuerschaltungen. (Penfold)
DM 12.80 ISBN 3-7723-1851-7
Jede der 33 Schaltungen wird sehr ausführlich, Bauelement für Bauelement – und dessen Funktion in der Schaltung – beschrieben.

RPB 186
MOS-Leistungstransistortechnik. Aufbau – Schaltungen – Anwendungen. (Schreiber)
DM 12.80 ISBN 3-7723-1861-4
Der Praktiker weiß mit Hilfe dieses Bandes künftig MOS-Leistungstransistoren vorteilhaft einzusetzen und erfolgreich anzuwenden. Ihm werden rund vierzig Schaltungen geboten, mit denen er auch sogleich etwas anfangen kann.

RPB 135
ABC der Mikroprozessoren und Mikrocomputer. Neue Fachwörter und Abkürzungen für Elektroniker, Programmierer und Praktiker verständlich gemacht. 2., neubearbeitete Auflage. (Pelka)
DM 12.80 ISBN 3-7723-1352-3

RPB 158
Sensible Sensoren. Elektronische Meßwertaufnehmer – Prinzipien und Anwendungsbeispiele. (Limann)
DM 9.80 ISBN 3-7723-1581-X
Diese Einführung gibt einen Überblick, welche Sensoren in Frage kommen, um Meß-, Steuer- und Regelungen aller Art elektronisch auszurüsten.

RPB 175
Infrarot-Elektronik. Eine Einführung in die Infrarottechnik mit Hobbyschaltungen und Experimenten. (Schreiber)
DM 12.80 ISBN 3-7723-1752-9
Die Bauteile sind preiswert, die Anwendungsmöglichkeiten sind vielfältig und neuartig. Was Infrarot ist, wie es erzeugt, gemessen, verarbeitet und angewendet wird, das wird hier erklärt.

Für alle Spurweiten einsetzbar.

RPB 192
Modellbahn-Mehrzugbetrieb. Erprobte Schaltungen mit selektiver Loksteuerung. (Meyer)
DM 12.80 ISBN 3-7723-1921-1
Über 15 Zugeinheiten können unabhängig voneinander in Fahrtrichtung und Fahrtgeschwindigkeit betriebssicher gesteuert werden. Das Geheimnis dieses Systems – die Schaltungen dazu – werden in diesem Band ausführlich beschrieben. Dazu gibt es Layouts und Bestückungspläne. Eingesetzt werden vorwiegend ICs.

RPB
electronic-taschenbücher bieten die Summe des Elektronikwissens für Beruf und Hobby.

RPB 154
KW-Amateurbildfunk SSTV und FAX. Technische Grundlagen – Nachbaupraxis – Betriebstechnik. (Pietsch)
DM 12.80 ISBN 3-7723-1541-0
Die bis ins Detail beschriebenen Schaltungen sind erprobt und regen zum Nachbau an.

RPB 6
Antennen für Rundfunk- und Fernseh-Empfang. Theoretische Überlegungen und gebräuchliche Ausführungsformen. (Mende)
DM 9.80 ISBN 3-7723-0066-9
Hier werden die Grundlagen der Antennentechnik behandelt.

RPB 65
Operationsverstärker-Anwendungen. Ein Wegweiser zur Verwirklichung eigener Ideen. (Hirschmann)
DM 12.80 ISBN 3-7723-0654-3
Der Ausgangspunkt aller Überlegungen ist die Optimierung von zwei Verstärker-Grundschaltungen. Danach wird ihre Anwendung in linearen und nichtlinearen Schaltungen, in aktiven Filtern, in Digitalschaltungen und in Signalgeneratoren gezeigt.

RPB 51
Kleine Fernseh-Bildfehler-Fibel. Typische Bildfehler, an Schirmbildaufnahmen erklärt. (Gies/Kirsch)
DM 9.80 ISBN 3-7723-0513-X
Reparaturzeiten und -kosten werden gespart. Gerade die kleinen, oft so schwer zu findenden Fehler bringt der Band ans Tageslicht. 68 Schirmbilder zeigen die häufigsten Fehlerquellen an.

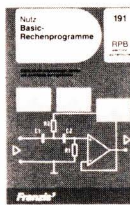
RPB 157
Meßgeräte und Meßverfahren für den Funkamateur
Auch einfache Meßgeräte bringen genügend genaue Meßergebnisse. (Link)
DM 9.80 ISBN 3-7723-1573-9
Schwierige Meßvorgänge mit einfachen Geräten und hinreichender Genauigkeit werden dem Funkamateur dargelegt.

RPB 112
Das Löten für den Praktiker. Beherzigenswerte Regeln für den Anfänger – nützliches Grundwissen für den Profi. (Strauss)
DM 12.80 ISBN 3-7723-1121-9
Der Autor kennt fast alle Tricks und nimmt dem Bastler die Furcht vor dem Löten.

RPB 137
Meßgeräte mit IC's. Erprobte Schaltungsvorlagen zum Selbstbau vielseitig verwendbarer Meßgeräte. (Sehrig)
DM 9.80 ISBN 3-7723-1371-X
Außer der eigentlichen Aufbaubeschreibung, die auch gedruckte Schaltungen umfaßt, wird auch die Funktionsweise der verwendeten ICs beschrieben. – Einige Schaltungen (Netzgerät, Digitalvoltmeter, Kalibriergenerator) eignen sich zum Nachbau durch Anfänger, während andere Schaltungen für Elektroniker mit einiger Selbstbauerfahrung gedacht sind.

RPB 167
Diavertonung. Regie und Technik der elektronisch gesteuerten Tonbildschau. (Tollmien)
DM 12.80 ISBN 3-7723-1671-9
Die Regieanweisungen gehen bis ins kleinste Detail. Die Elektronikrezepte bis hin zur Printplatte und Stückliste bringen den Hobby-Fotografen echt weiter.

RPB 136
Transistorisierte Netzgeräte
Spannung und Strom geregelt durch Halbleiter. (Strobel)
DM 6.80 ISBN 3-7723-1366-3
Die in dem vorliegenden Buch beschriebenen Schaltungen sind einfach und klar verständlich gehalten, um auch dem Anfänger – ob Techniker oder Amateur – wertvolle Hinweise und Tips für den Bau seiner Netzgeräte zu vermitteln.



Die Computer-Schaltungen schnell erlernen.

RPB 191
Basic-Rechenprogramme. Elektronik-Grundschaltungen schnell und zuverlässig durchgerechnet. (Nutz)
DM 9.80 ISBN 3-7723-1911-4
Enthalten sind 19 Programme, die auf jedem Basic-Computer lauffähig sind. Mit diesen Programmen kann jeder seine Schaltungen schnell berechnen, denn die mathematische Routinearbeit entfällt.

RPB 159
Die logisch gesteuerte Modelleisenbahn. Eine Großanlage wird mit neuartigen Bauelementen und Schaltungen sowie mit Mikroprozessoren durchautomatisiert. (Platerink)
DM 12.80 ISBN 3-7723-1591-7

RPB 169
Kleiner Basic-Wortschatz. Die wichtigsten Basic-Begriffe einfach erklärt und gelistet. (Busch)
DM 12.80 ISBN 3-7723-1692-1
Was in den maschinenorientierten Handbüchern verstreut, schier systemlos untergebracht ist, findet sich hier fein säuberlich, exakt, präzise und genau in einem zuverlässigen Lexikon.

RPB 50
Praktischer Antennenbau. Ein Ratgeber für Entwurf und Ausführung von Antennenanlagen aller Rundfunkwellenbereiche. (Mende)
DM 12.80 ISBN 3-7723-0509-1
Der Band befaßt sich mit Rundfunk- und Fernsehantennen für alle Wellenbereiche und Programme und enthält die Abmessungen der Antennen für alle Kanäle.

RPB 64
Einführung in die Operationsverstärker-Technik. Ein Wegweiser, Aufbau, Arbeitsweise und Eigenschaften der Operationsverstärker besser zu verstehen. (Hirschmann)
DM 9.80 ISBN 3-7723-0643-8
Von besonderem Interesse für den Praktiker sind die vielen Beispiele der äußeren Beschaltungsarten, die Hinweise für die erschöpfende Auswertung der Datenblätter und die Tips für das Auslegen der Schaltungen.

RPB 84
Fernsehantennen-Praxis. Ein zuverlässiger Leitfaden zum besten Fernsehempfang. (Mende)
DM 9.80 ISBN 3-7723-0844-9
Das ist ein grundsätzliches Kompendium der FA-Antennen-Technik, das getrost als „Leitfaden zum besten Fernsehempfang" bezeichnet werden kann.



Preiswerte Peilantenne durch Selbstbau.

RPB 182
Aktive Antennen für DX-Empfang. Theorie – Selbstbau – Praxis. (Best)
DM 9.80 ISBN 3-7723-1821-5
Rat und Hilfe, die Möglichkeiten einer aktiven Antenne voll auszunutzen, findet der DXer in diesem Band.

RPB 172
FET-Theorie. Von den theoretischen Grundlagen zur praktischen Schaltungstechnik der Feldeffekt-Transistoren. (Dielemann)
DM 9.80 ISBN 3-7723-1721-9
Der FET soll effektiv eingesetzt werden. In der diskreten Bauweise und in der Großintegration. Die Kenntnisse dazu vermittelt dieser Band.

RPB 146
Halbleiterspeicher. Eine Kurz-Darstellung der Halbleiterspeicher von den Grundlagen bis zur Anwendung. (Bonerz)
DM 9.80 ISBN 3-7723-1461-9
Der Techniker findet einen Überblick der Typenvielfalt, lernt den einzelnen Speicher richtig zu beurteilen und optimal einzusetzen.

RPB 165
BIFET – BIMOS – CMOS in Feldeffekt-Operationsverstärkern. Die Eigenschaften der FET-Operationsverstärker und ihre zahlreichen Anwendungsmöglichkeiten. (Schreiber)
DM 12.80 ISBN 3-7723-1651-4
Die Anwendung von FET-Operationsverstärkern ist durch die neuen Technologien (BIFET, BIMOS, CMOS) in jeder Beziehung interessanter geworden. Die technische Komponente behandelt dieser Band.



Rund 120 RPB electronic-taschenbücher bieten Ihnen Informationen, Tips und Ratschläge. Diese Anzeige zeigt lediglich eine Auswahl. Bitte fordern Sie das kostenlose Gesamtverzeichnis unter der Bestellnummer P 277 an.
Franzis-Verlag, Postf. 31 01 20, 8000 München 37.

Franzis-Bücher erhalten Sie durch jede Buchhandlung sowie in den einschlägigen Fachhandlungen. Bestellungen auch an den Verlag. (85)

8-Bit-Prozessor bietet 32-Bit-Architektur

Mit der 8-Bit-Version des Mikroprozessors MC68000 stehen in der 68000-Familie auch für kleine Mikrocomputersysteme die architektonischen Merkmale von Großrechnern zur Verfügung. Konzipiert wurde der Prozessor MC68008 für kosteneffektive Systeme mit 8-Bit-Datenbussen, bei denen nicht auf die Vorteile einer 32-Bit-Mikroprozessor-Architektur verzichtet werden soll. Die Leistung dieses Prozessors liegt über der aller bekannten 8-Bit-Typen, sie übertrifft sogar einige 16-Bit-CPUs.

Die Zentraleinheit MC68008 ist vollständig codekompatibel mit dem Typ MC68000. Damit sind Programme, die ursprünglich für den Typ 68000 entwickelt wurden, auch auf der 8-Bit-Version lauffähig und umgekehrt. Diese Aussage gilt sowohl für den Quellen- als auch Objekt-Code. Eine Systemimplementierung, die auf einem 8-Bit-Datenbus basiert, reduziert die Kosten im Vergleich zu 16-Bit-Systemen, weil die vorhandenen Bauelemente wirtschaftlicher ausgenutzt werden. Speicher und Peripheriebausteine mit Wortbreite tragen zur Wirtschaftlichkeit bei. Außerdem machen die multiplexfreien Daten- und Adreßbusse externe Demultiplexer überflüssig, wodurch sich der Systemaufbau vereinfacht. Die Architektur des MC68008 entspricht vielen Anforderungen verschiedener Anwendungen. Sie bietet dem Benutzer 17 Register mit einer Breite von jeweils 32 Bit, einen Satz von 56 Grund-Befehlen, wodurch eine effiziente Hardware-Systemimplementierung ermöglicht wird und sich hohe Verarbeitungsleistung ergibt.

1 Registerstruktur

Wie aus *Bild 1* hervorgeht, entspricht das Programmier-Modell des MC68008 dem des MC68000. Es umfaßt 17 Register mit einer Länge von 32 Bit, einen 32-Bit-Programmzähler und ein 16-Bit-Statusregister. Die ersten acht Register (D0...D7) werden als Datenregister für Byte- (8 Bit), Wort- (16 Bit) und Lang-Wort-Operationen (32 Bit) benutzt. Der zweite Satz von sieben Registern (A0...A6) sowie der System-Stack-Pointer (A7) kann als Software-Stapel-Zeiger- und Basis-Adreß-Register Verwendung finden. Außerdem eignen sich diese Register für Wort- und Lang-Wort-Operationen. Jedes dieser siebzehn Register kann außerdem auch als Index-Register benutzt werden.

Der System-Stack wird im Zusammenhang mit verschiedenen Befehlen benutzt. 14 Adressierungsarten ermöglichen den Aufbau eines Benutzer-Stacks und von Warteschlangen. Beim System-Stack-Pointer handelt es sich in Abhängigkeit vom Wert des S-Bits im Status-Register entweder um den übergeordneten Stapel-Zeiger (*Supervisor Stack Pointer* – SSP) oder den Benutzer-Stapel-Zeiger (*User Stack Pointer* – USP). Wenn dieses Bit gesetzt ist, zeigt es an, daß der Prozessor im Supervisor-Zustand arbeitet, dann ist der USP nicht in Benut-

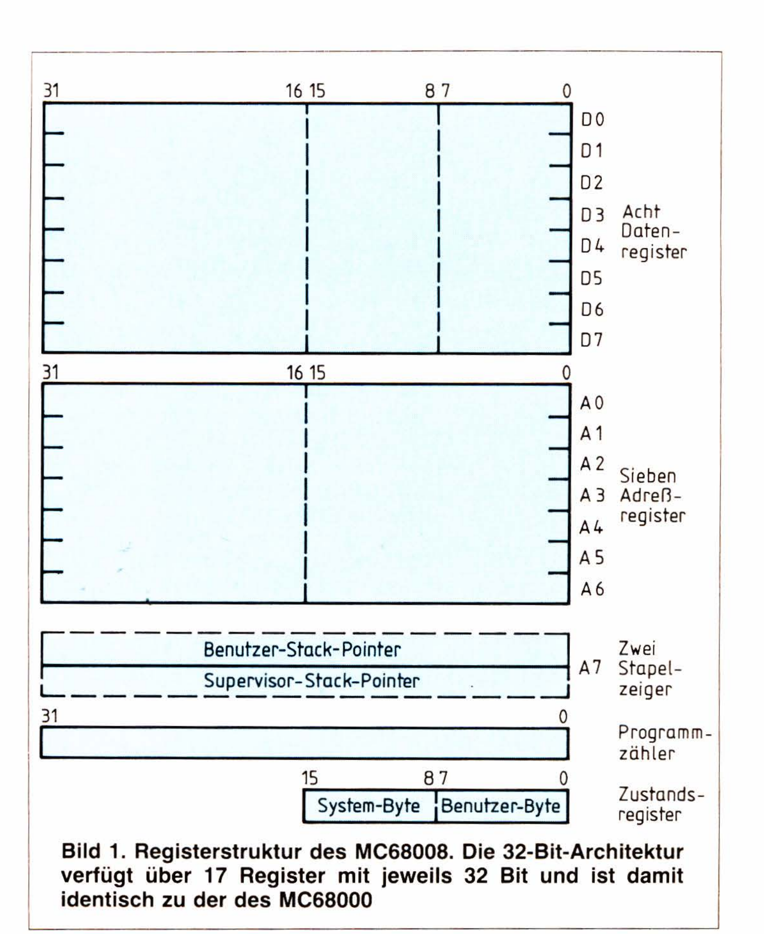


Bild 1. Registerstruktur des MC68008. Die 32-Bit-Architektur verfügt über 17 Register mit jeweils 32 Bit und ist damit identisch zu der des MC68000

zung. Wenn das S-Bit sich im Low-Zustand befindet, zeigt es an, daß der Prozessor im Benutzer-Zustand arbeitet, wobei der USP aktiv ist und der SSP vor Modifikationen durch den Benutzer geschützt wird.

Im großen nichtsegmentierten linearen Adreßbereich des MC68008 können umfangreiche modulare Programme entwickelt und wirtschaftlich ausgeführt werden. Ein solcher Adreßbereich ermöglicht es, daß die Segment-Abmessungen von den Erfordernissen der Anwendungen bestimmt werden. Der Entwickler ist nicht gezwungen, vorgegebene Segmente berücksichtigen zu müssen.

Tabelle 1. Befehlssatz des Prozessors MC68008

Mnemonischer Code	Beschreibung
ABCD	Dezimal-Addition mit Erweiterung
ADD	Addition
AND	Logisches UND
ASL	Arithmetik-Verschiebung, links
ASR	Arithmetik-Verschiebung, rechts
B_{CC}	Bedingte Verzweigung
BCHG	Bit-Test und -Veränderung
BCLR	Bit-Test und -Rücksetzen
BRA	Verzweigung
BSET	Bit-Test und -Setzen
BSR	Verzweigung zum Unterprogramm
BTST	Bit-Test
CHK	Überprüfen der Register auf Begrenzung
CLR	Rücksetzen des Operanden
CMP	Vergleich
DB_{CC}	Test-Bedingung, Dekrementierung und Verzweigung
DIVS	Division mit Vorzeichen
DIVU	Division ohne Vorzeichen
EOR	Exklusiv-ODER
EXG	Register-Austausch
EXT	Vorzeichen-Erweiterung
JMP	Sprung
JSR	Sprung zum Unterprogramm
LEA	Laden der effektiven Adresse
LINK	Binden des Stacks
LSL	Logische Verschiebung, links
LSR	Logische Verschiebung, rechts
MOVE	Transfer
MOVEM	Transfer mehrerer Register
MOVEP	Transfer peripherer Daten
MULS	Multiplikation mit Vorzeichen
MULU	Multiplikation ohne Vorzeichen
NBCD	Negierung einer Dezimalzahl mit Erweiterung
NEG	Invertierung
NOP	Keine Operation
NOT	Einer-Komplement
OR	Logisches ODER
PEA	Transfer der effektiven Adresse
RESET	Zurücksetzen externer Elemente
ROL	Rotation, links ohne Erweiterung
ROR	Rotation, rechts ohne Erweiterung
ROXL	Rotation, links mit Erweiterung
ROXR	Rotation, rechts mit Erweiterung
RTE	Rückkehr aus Sonder-Funktion
RTR	Rückkehr und Wiederherstellung des Zustandes
RTS	Rückkehr vom Unterprogramm
SBCD	Dezimal-Subtraktion mit Erweiterung
S_{CC}	Bedingtes Setzen
STOP	Stop
SUB	Subtraktion
SWAP	Umtauschen der Daten-Register-Hälften
TAS	Testen und Setzen des Operanden
TRAP	Trap
TRAPV	Trap bei Überlauf
TST	Test
UNLK	Bindung aufheben

Das in *Bild 2* dargestellte Status-Register besteht aus zwei Bytes: Benutzer- und System-Byte. Im Benutzer-Byte sind 5 Bit enthalten, die die Bedingungs-Codes für Überlauf (V), Null (Z), Negativ (N), Übertrag (C) sowie erweitert (X) definieren. Das System-Byte enthält 5 Bits. Drei dieser Bits werden zur Definition der laufenden Interrupt-Priorität benutzt; jede Interrupt-Ebene, die oberhalb der gerade bearbeiteten Masken-Ebene liegt, wird erkannt. Die Interrupts der Ebene 7 sind nicht maskierbar, das heißt, daß diese jederzeit verarbeitet werden. Zwei zusätzliche Bits zeigen an, ob der Prozessor im „Trace-Mode" (T) oder im Supervisor-Zustand (S) arbeitet. Nicht benutzte Status-Register dienen späteren Erweiterungen der M68000-Familie.

2 Befehlssatz

Der 8-Bit-Mikroprozessor MC68008 ist vollständig codekompatibel mit dem Typ MC68000. Das bedeutet, daß Programme, die für den Typ MC68000 entwickelt wurden, auch auf dem MC68008 ablaufen und umgekehrt. Dies gilt für den Quellen- oder Objekt-Code. Entworfen wurde der Befehlssatz unter dem Gesichtspunkt, daß die Anzahl der mnemonischen Instruktionen, die sich der Programmierer merken muß, möglichst gering ist. Zur weiteren Erleichterung der Programmierung sind die Adressierungsarten orthogonal ausgelegt.

Tabelle 1 zeigt den Befehlssatz, der als Programmier-Werkzeug alle Prozessor-Funktionen umfaßt, die notwendig sind, um Datentransport, Ganzzahl-Arithmetik, logische Operationen, Verschiebe- und Rotations-Vorgänge, Bit-Manipulation, BCD-Operationen sowie Programm- und System-Steuerungen auszuführen. Zusätzliche Befehle sind Varianten oder Unterbefehle davon (*Tabelle 2*).

Jede Instruktion arbeitet mit wenigen Ausnahmen mit Bytes, Worten und Lang-Worten. Die Befehle besitzen eine Länge, die zwischen einem und fünf Worten liegt. Festgelegt werden die Länge der Instruktion und der Operation, die ausgeführt werden soll, durch das erste Wort des Befehls. Dieses trägt den Namen „Operations-Wort". Der Rest des Wortes spezifiziert die Operanden. Es handelt sich entweder um direkte Operanden oder Erweiterungen der Adressierungsart, die im Operations-Wort spezifiziert ist.

Die meisten Befehle können jede der 14 Adressierungsarten benutzen, die in *Tabelle 3* zusammengefaßt sind. Diese Adressierungsarten bestehen aus sechs grundlegenden Typen: Register-Direkt, Absolut, bezogen auf den Programmzähler, Register-Indirekt, Immediate sowie Implizit. Die registerindirekten Adressierungsarten besitzen außerdem Einrichtungen zur Nach-

und Vor-Inkrementierung, zur Erzeugung eines Offset sowie zur Indizierung. Die Adressierungsart, die sich auf den Programmzähler bezieht, kann in Kombination mit der Indizierung und Offset-Erzeugung beim Schreiben relokatibler Programme benutzt werden.

Durch Kombination von Befehls- und Datentypen sowie Adressierungsarten ergeben sich mehr als 1000 nützliche Instruktionen. Dazu zählen unter anderem Multiplikation und Division mit oder ohne Vorzeichen, schnelle Arithmetik-Operationen, BCD-Arithmetik sowie erweiterte Operationen (über „Traps").

3 Hardware-Implementierung

Im Mikroprozessor MC68008 sind verschiedene innovative Konzepte zusammengefaßt. Kommunikation mit dem Speicher erfolgt multiplexfrei und asynchron über einen Hochleistungs-Adreß- und Daten-Bus (die E/A-Funktionen sind im Speicherbereich abgebildet). Vereinfacht wurde die Konzipierung des MC68008 durch den Mikrocode, der die Möglichkeit zur Erweiterung des Prozessors offenläßt. Verarbeitung von Fehler-Funktionen trägt dazu bei, die Integrität der Maschine aufrecht zu erhalten. *Bild 3* zeigt die externen Anschlüsse des Mikroprozessors MC68008.

Durch asynchrone Busstruktur wird hohe Flexibilität bei der Auswahl der für die Anwendung erforderlichen Zugriffszeit erreicht, wodurch Kosten und Leistung zueinander in einem ausgewogenen Verhältnis stehen. Ein multiplexfreier Bus wird verwendet, weil dieser einen um 30 % höheren Durchsatz gegenüber einem Multiplex-Adressen- und Daten-Bus aufweist.

Zum Anschluß von Peripherieeinheiten wurden die gleichen Steuersignale wie die des Typs M6800 verwendet. Es passen die synchronen Peripherieeinheiten der M68000-Familie an die asynchronen Anschlüsse MC68008. Das Enable-Signal (E) wird vom MC68008 über einen Zeitraum von zehn Taktperioden (sechs Takte Low, vier Takte High) erzeugt. „Peripherie-Adresse gültig" ($\overline{VPA}$ – *Valid Peripheral Address*) ist ein Eingangssignal für den MC68008, das von dem Decodierer erzeugt wird, der die Chip-Select-Signale der M6800-Peripherieeinheiten bereitstellt. Nachdem das Signal

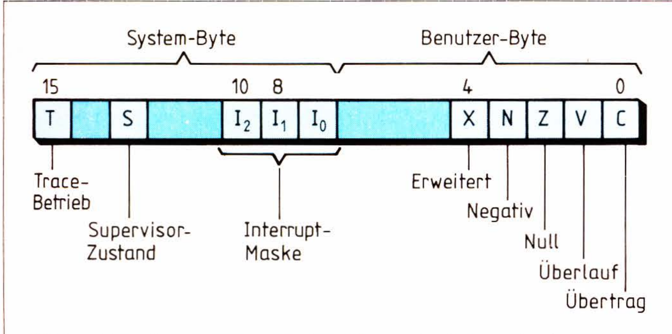


Bild 2. Das Statusregister des MC68008. Die nicht benutzten Bits sind für spätere Erweiterungen freigehalten

Tabelle 2. Versionen der Befehle

Befehlstyp	Version	Beschreibung
ADD	ADD	Addition
	ADDA	Addition einer Adresse
	ADDQ	Schnelle Addition
	ADDI	Direkte Addition
	ADDX	Addition mit Erweiterung
AND	AND	Logisches UND
	ANDI	Direkte UND-Funktion
CMP	CMP	Vergleich
	CMPA	Vergleich einer Adresse
	CMPM	Vergleich eines Speicherinhaltes
	CMPI	Direkter Vergleich
EOR	EOR	Exklusiv-ODER
	EORI	Direktes Exklusiv-ODER
MOVE	MOVE	Transfer
	MOVEA	Transfer einer Adresse
	MOVEQ	Schneller Transfer
	MOVE from SR	Transfer vom Status-Register
	MOVE to SR	Transfer zum Status-Register
	MOVE to CCR	Transfer zu den Bedingungs-Codes
	MOVE USP	Transfer des Benutzer-Stapel-Zeigers
NEG	NEG	Inversion
	NEGX	Inversion mit Erweiterung
OR	OR	Logisches ODER
	ORI	Direktes ODER
SUB	SUB	Subtraktion
	SUBA	Subtraktion einer Adresse
	SUBI	Direktes Subtahieren
	SUBQ	Schnelle Subtraktion
	SUBX	Subtraktion mit Erweiterungen

$\overline{VPA}$ anliegt, synchronisiert die MPU MC68008 ihren Buszyklus auf E und gibt das Signal $\overline{VMA}$ (*Valid Memory Address*) aus. $\overline{VMA}$ sollte dazu benutzt werden, sicherzustellen, daß der M6800-Peripherie-Chip-Select im erforderlichen Zeitrahmen liegt. In Abhängigkeit des Zeitpunktes, bei dem $\overline{VPA}$ in bezug auf E angelegt wurde, können die synchronen Buszyklen zwischen 8 und 18 Taktperioden dauern.

Die Prozessor-Zustandsausgänge zeigen den Zustand (Benutzer oder Supervisor) sowie die Art des gerade erfolgten Zugriffes an (Programm oder Daten). Außerdem benutzt der Prozessor die Status-Ausgänge zur Anzeige der Ausführung eines Interrupt-Acknowledge-Zyklus. Bus-Zuteilungs-Steuerung wird durch Benutzung von zwei Leitungs-Interfaces ausgeführt, die in Form der Signale „*Bus Request*" ($\overline{BR}$) und „*Bus Grant*" ($\overline{BG}$) vorliegen. In einem System, bei dem mehrere Prozessoren oder DMA-Einheiten am Bus angeschlossen sind, legen diese Signale fest, welche Einheit der Bus-Master ist. Für erweiterte Zuteilungs-Funktionen und Unterstützung von acht Bus-Mastern kann der Bus-Arbitrations-Baustein MC68452 aus der M68000-Familie Verwendung finden.

Drei Interrupt-Steuereingänge zeigen die decodierte Prioritätsebene der Einheit an, die einen Interrupt auslö-

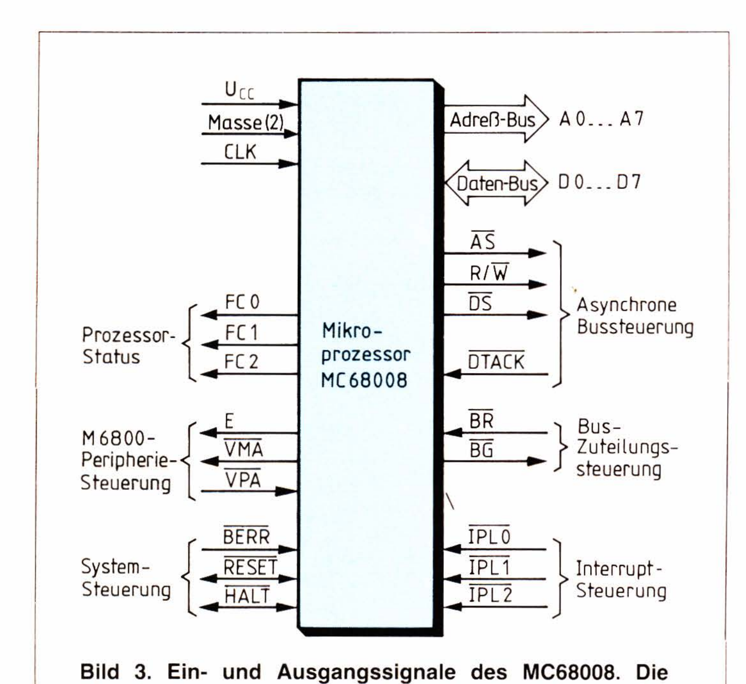


Bild 3. Ein- und Ausgangssignale des MC68008. Die Signale sind so ausgelegt, daß Interfacebausteine aus der M68000-Familie angeschlossen werden können

Tabelle 3. Adressierungsarten des Prozessors MC68008

Register-Direkt Daten-Register-Direkt Adressen-Register-Direkt
Absolut (Daten) Absolut, Kurz Absolut, Lang
Programm-Zähler Relativ mit Offset Relativ mit Index und Offset
Register-Indirekt Register-Indirekt mit Nach-Inkrementierung Register-Indirekt mit Vor-Inkrementierung Register-Indirekt mit Offset Register-Indirekt mit Offset, indiziert
Direkte Daten-Adressierung Direkt Schnell
Implizit Implizite Register-Adressierung

sen will. Ebene 7 besitzt höchste Priorität und ist nicht maskierbar, während Ebene 0 zeigt, daß jeder Interrupt akzeptiert wird. Die drei System-Steuereingänge „RESET", „HALT" sowie „BERR" werden zum Zurücksetzen des Prozessors und/oder des Systems, zum Anhalten des Prozessors oder Anzeige, daß ein Busfehler aufgetreten ist, benutzt. Die HALT-Leitung kann auch vom Prozessor zur Anzeige von fatalen System-Fehlern verwendet werden, um System-Integrität zu gewährleisten.

4 Verarbeitung von Sonderzuständen

Verarbeitung von Sonderzuständen ermöglicht der MCU MC68008 die Behandlung von Interrupts, Adressierungsfehlern, nichtimplementierten Befehlen und anderen problematischen Zuständen, während absolute Systemintegrität aufrecht erhalten wird. Der „*Exception-Processing*"-Zustand hängt mit Interrupts, Trap-Instruktionen, Tracing und anderen besonderen Konditionen zusammen. Ein Sonderzustand kann intern durch eine Instruktion oder durch ungewöhnliche Bedingungen, die während der Programmausführung auftreten, erzeugt werden. Die Verarbeitung verfügt über eine effiziente Kontext-Umschaltung, wodurch die System-Integrität gewährleistet wird.

Mehr MPUs für 68000-Familie

Neben der hier beschriebenen 32-Bit-Zentraleinheit mit 8-Bit-Bus (68008) und dem „Grundmodell" mit 16-Bit-Bus (68000) sind weitere MPUs verfügbar. Bei dem Typ MC68020 handelt es sich um einen „echten" 32-Bit-Prozessor, der neben dem 68000-Befehlssatz auch beispielsweise Fließkommaoperationen und Coprozessoren unterstützt.

Der MCU-Baustein MC68010 ist eine 16/32-Bit-CPU, die insbesondere für virtuelle Speicherarchitekturen konzipiert ist. Die ELEKTRONIK wird demnächst auch über diese beiden Bausteine ausführlich berichten.

5 Anwendungen

Mit dem Baustein MC68008 eröffnet sich ein einfacher Weg, hohe Leistung in eine System-Umgebung, die über 8-Bit-Datenpfade verfügt, zu implementieren, gleichzeitig aber das Wachstum bis auf eine 32-Bit-Konfiguration offenzuhalten. Aufgrund der Kompatibilität in Soft- und Hardware ist das leicht möglich. Der Mikroprozessor MC68008 weist etwa 60 % der Leistung einer 16-Bit-MPU vom Typ 68000 auf. Dieses ist für viele Anwendungen durchaus ausreichend und stellt ein optimales Preis/Leistungs-Verhältnis dar.

Nach Unterlagen der Firma Motorola, bearbeitet von *P. von Bechen*.

Literatur

[1] *Daniels, G., Lösel, M.*: Ein 16-Bit-Mikroprozessor in HMOS-Technik. ELEKTRONIK 1979, H. 10, S. 47...53.

CPU der 68000-Familie unterstützt virtuellen Speicher

Die 16/32-Bit-Mikroprozessorfamilie MC68000 wurde um eine CPU erweitert, die die Typenbezeichnung MC68010 trägt und Einrichtungen besitzt, durch die sie für die Verwendung in Systemen mit virtuellen Speichern geeignet ist. Wie auch die anderen Mitglieder der 68000-Familie ist auch dieser Baustein vollständig kompatibel zu den übrigen Typen. Einige der wichtigen Unterschiede zu den anderen Familienmitgliedern betreffen den Betrieb des Prozessors, wenn ein Fehler im Buszyklus auftritt. Darüber hinaus bietet der Mikroprozessor MC68010 bessere Unterstützung eines Betriebssystems.

Die wohl wichtigste Verbesserung des Mikroprozessors MC68010 gegenüber den übrigen Mitgliedern der 68000-Familie ist dessen Einrichtung für vollständige Speicherung des internen Prozessorzustandes beim Auftreten eines Bus-Fehlersignals. Nach Behebung des Fehlers kann der ursprüngliche Zustand wieder hergestellt und der laufende Betrieb aufgenommen werden. Darüber hinaus kann ein MC68010 als virtueller Prozessor mit einem übergeordneten Betriebssystem verwendet werden, unter dem untergeordnete Betriebssysteme angeordnet sind. Damit werden auch virtuelle Ein-/Ausgabevorgänge unterstützt. Der dritte wichtige Unterschied ist die Möglichkeit zu einem verzögerten Bus-Fehlersignal zum Abbruch des Buszyklus. Diese Einrichtung trägt die Bezeichnung „Bus-Relaxation" und sorgt dafür, daß Fehlererkennung und -Korrektur (EDAC) mit dem Mikroprozessor MC68010 ausgeführt werden können, ohne die Ausführungszeit der Befehle im fehlerfreien Betrieb zu beeinflussen.

Die wichtigsten Merkmale des Mikroprozessors MC68010 sind im folgenden kurz zusammengefaßt:

- 32-Bit-Daten- und Adreß-Register
- linearer Adressierbereich von 16 MByte
- 58 leistungsfähige Befehlstypen
- Verarbeitung von fünf Haupt-Datentypen
- Memory-Mapped-E/A
- 8-, 16- und 32-Bit-Operationen
- 14 Adressierungsarten
- Unterstützung virtueller Speicher
- Funktion als virtuelle Maschine
- relokatible Vektortabelle
- übergeordneter Zugriff auf Benutzer-Ressourcen
- Kompatibilität mit anderen Mitgliedern der 68000-Familie.

1 Architektur

Bild 1 zeigt, daß beim MC68010 dem Benutzer 16 Register mit 32 Bit neben dem 32-Bit-Programmzähler und dem 8-Bit-Zustandscoderegister zur Verfügung stehen. Die ersten acht Register (D0...D7) werden als Datenregister für Byte- (8 Bit), Wort- (16 Bit) und Langwort-Datenoperationen (32 Bit) benutzt. Der zweite Satz von sieben Registern (A0...A6) sowie der Benutzer-Stackpointer (A7) können als Software-Stackpointer sowie als Basis-Adreßregister Verwendung finden. Alle diese Register können als Indexregister benutzt werden.

Das Supervisor-Programmiermodell bietet zusätzliche, geschützte Ressourcen (*Bild 2*). Steuerbits in dem oberen Bereich des Statusregisters ermöglichen es einem Programm, einen Befehls-Trace durchzuführen oder nicht, auf die Benutzerprogrammebene überzugehen oder Interrupts geringer Priorität zu maskieren. Neben einem 32-Bit-Supervisor-Stackpointer ist auch noch ein neues 32-Bit-Vektor-Basisregister vorgesehen.

Zwei zusätzliche Funktionscode-Register werden für spezielle Befehle benötigt.

Die Ausführung von Befehlen im MC68010 findet auf zwei privilegierten Ebenen statt. Auf der Benutzerebene sollten Hochsprachen, Anwendungssoftware und Dienstprogramme laufen. Betriebssysteme, Interruptverwaltung sowie Ressource- und System-Steuerfunktionen sollten auf der oberen Supervisor-Ebene ausgeführt werden. Einige Instruktionen und Ressourcen sind privilegiert und aus diesem Grund nicht für Operationen auf der Benutzerebene verfügbar.

Für die Programmsteuerung besitzt der Mikroprozessor MC68010 spezielle Befehle, z. B. für bedingte Sprünge und Schleifen. Viele Formen von Datenverschiebeoperationen sind ebenfalls möglich, wodurch

Transfers zu und von Registern, Stacks, Warteschlangen, Pointern, Speichern, E/A-Einheiten sowie den Status- und Programmzähler-Registern möglich sind. Arithmetische und logische Operationen mit Wortbreiten von 8, 16 und 32 Bit lassen als Bestimmungswort sowohl Speicher als auch Register zu. Arithmetikoperationen unterschiedlicher Wortlänge oder verschiedener Genauigkeitsstufen werden durch speziell dafür konzipierte Instruktionen erleichtert. Schiebe- und Rotations-Befehle erlauben mehrfache serielle Schiebeoperationen mit maximal 32 Bit. Für BCD-Arithmetik sind ebenfalls spezielle Befehle vorgesehen. Einzelne Bits, die sich entweder in einem Register oder im Speicher befinden, können durch Bitmanipulationsbefehle getestet und modifiziert werden. Höhere Programmiersprachen und anspruchsvolle Programmierung wird mit Hilfe leistungsfähiger Befehle besser unterstützt, z. B. durch einen „unsichtbaren" Befehl für Multiprozessor-Ressource-Zuweisung.

Daten in Registern können direkt adressiert werden; auf Daten im Speicher kann man mit jeder Instruktion auf sehr unterschiedliche Weise zugreifen. Auf Daten im Speicher kann man wie auf einen Stack oder eine Warteschleife zugreifen, über ein Pointerregister, dynamisch oder statisch indiziert, relativ auf den Programmzähler bezogen oder durch Finden der absoluten Adresse.

Interrupts und Traps ermöglichen die Unterbrechung des normalen Instruktionsflusses beim Auftreten von externen oder internen Ereignissen, die bedient werden müssen. Den Interrupts kann einer von sieben Prioritäten zugewiesen werden, wobei Interrupts geringerer Priorität durch Maskierung zurückzustellen sind. Sie können durch einen Vektor an die richtige Service-Routine verwiesen werden. Traps und Interrupts können auch durch Exception-Handler bedient werden.

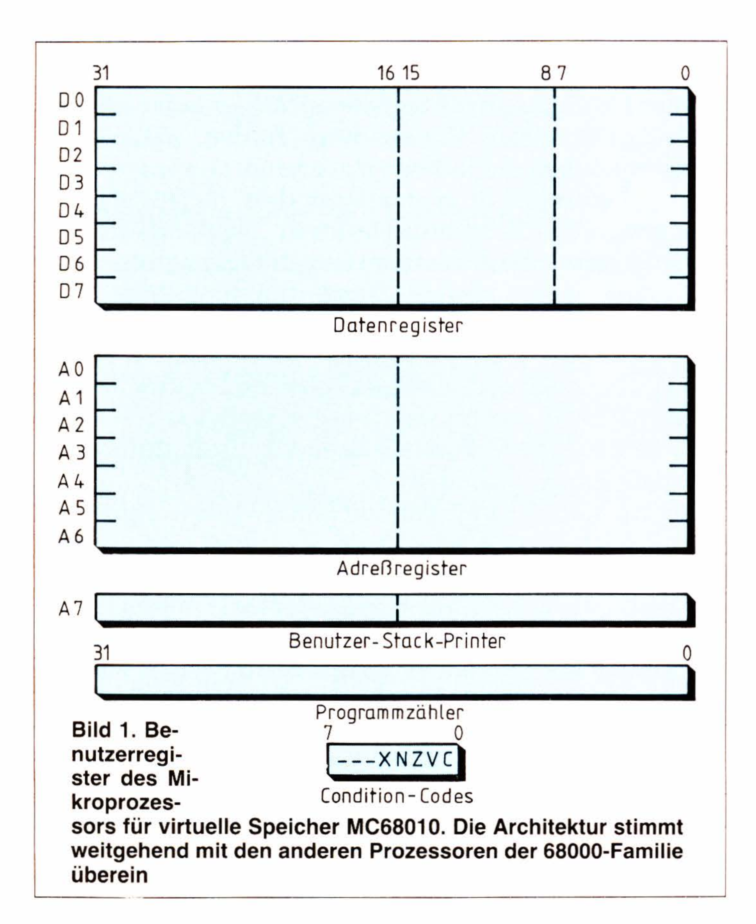


Bild 1. Benutzerregister des Mikroprozessors für virtuelle Speicher MC68010. Die Architektur stimmt weitgehend mit den anderen Prozessoren der 68000-Familie überein

Der Mikroprozessor MC68010 läßt sich innerhalb von Multiprozessorsystemen in einer Vielzahl von Konfigurationen anwenden. Unterstützt wird die Systemauslegung durch eine Busarbitrationsschaltung auf dem Chip, die mit einer 3poligen Busverbindung arbeitet. Sie bietet alle Einrichtungen zur Freigabe der Adreß-, Daten- und Steuerbusse, wenn andere Prozessoren diese benötigen.

Auf Speicher und E/A-Einheiten des MC68010 wird über einen asynchronen Datenbus mit 16 Bit Breite zugegriffen. Obwohl intern 32 Adreßbits benutzt werden, sind 24 davon nach außen in vier verschiedenen Adreßbereichen zugänglich, wobei jeder Bereich 16 MByte umfaßt. Der Adressierbereich ist vollkommen linear und weist nicht wie bei der Segmentierung künstliche Adressen auf, die nur mit Vorsicht überschritten werden können.

Der Mikroprozessor MC68010 ist anschlußkompatibel mit dem Typ MC68000. Lediglich der Zeitverlauf einiger Signale, z. B. BERR, ist leicht modifiziert. Eine vollständige Unterscheidung zwischen Benutzer- und Supervisor-Betriebsarten ergibt sich dadurch, daß der Befehl „MOVE SR, ‹ea›" zur privilegierten Instruktion gemacht wurde. Beim Mikroprozessor MC68000 erlaubt diese Instruktion den Zugriff auf die Supervisor-Information (T-, S- und I-Bits). Wenn der Befehl auf dem MC68010 im Benutzer-Betrieb verwendet wird, führt das zu einer Privilegverletzung. Der Exception-Handler überprüft, ob es sich um ein Anwendungsprogramm oder Betriebssystem handelt und kann den Befehl gegebenenfalls simulieren. Die gesamte Benutzersoftware, die für den MC68000 geschrieben ist, wird auf dem MC68010 in gleicher Weise ausgeführt. Betriebssystemcode auf privilegierter Ebene läuft auch auf dem MC68010 mit Ausnahme kleiner Erweiterungen, die notwendig sind, um den nun privilegierten Befehl „MOVE SR, ‹ea›" verarbeiten zu können, und des zusätzlichen Vektor-Nummern-Wortes, das dem Stack für die Exception-Verarbeitung hinzugefügt werden muß.

2 Unterstützung virtueller Speicher

Bis heute ist noch kein Mikroprozessor auf dem Markt, der ein Speichersystem „einhändig" unterstützt, wenn ein fehlerhafter Speicherzyklus auftritt. Um zu verhindern, daß fehlerhafte Daten in einer internen Operation benutzt oder Teile von Daten bei der Adressierung außerhalb des zulässigen Adreßbereiches verlorengehen, erfordern diese Prozessoren einen zweiten Prozessor auf dem Bus, der die Ursache des Fehlers erkennen und korrigierende Maßnahmen ergreifen muß. Ohne korrigierende Maßnahmen besteht die Möglichkeit zu permanentem Datenverlust.

Wenn beim Prozessor MC68010 ein Bus-Fehlersignal (BERR) auftritt, setzt dieser die Ausführung der laufen-

den Instruktion sofort aus und beginnt im Supervisor-Stack den gesamten internen Zustand des Prozessors abzulegen einschließlich der Registerinhalte und Prozeßinformationen. Es wird automatisch in eine Bus-Fehler-Serviceroutine eingesprungen. Die abgelegte Information eignet sich für den betreffenden Prozessor oder einen anderen Mikroprozessor vom Typ MC68010, um in den gleichen Zustand zurückzugelangen, der vor dem Auftreten des Fehlers vorlag, den vorher fehlerhaften Buszyklus abzuarbeiten und die Operation dort fortzuführen, wo sie angehalten wurde. All dies geschieht automatisch. Wenn der Mikroprozessor die RTE-Instruktion ausführt, wird der MC68010 mit dem vorher abgelegten Maschinenzustand neu geladen, worauf er die vorher zurückgestellte Instruktion weiter abarbeitet. Die Routine, die dann abläuft, ist funktionsmäßig exakt die gleiche wie eine ohne auftretenden Fehler. Lediglich die zusätzliche Zeit zur Ausführung der Exception-Routine muß berücksichtigt werden. Weil der Bus-Fehler-Handler vom Supervisor betrieben wird und der vorhergehende Zustand des Prozessors durch die Information, die im Stack geschützt ist, wiederhergestellt werden kann, wird die Routine vom auftretenden Fehler nicht beeinflußt.

Wenn ein Busfehler auftritt, empfängt der Supervisor-Stack eine Anzahl Bytes mit der Zustandsinformation. Im Prozessor für den virtuellen Speicher enthält der Stack die Registerinhalte von PC, SR, die Speicheradresse, die den Fehler verursachte, den Op-Code der Instruktion, die ausgeführt wurde, während der Fehler auftrat, den Funktionscode und weitere Bus-Steuerinformationen. Alle diese Informationen können zusätzlich mit extern verfügbaren Informationen (z. B. die ausgegebene physikalische Adresse der Speicherverwaltungseinheit MC68451) vom Betriebssystem dazu benutzt werden, die Ursache des Fehlers zu analysieren. Außerdem sind im Stack des MC68010 Informationen über den internen Zustand enthalten. Diese interne Information wird außer einer Angabe über die Menge der Information im Stack nicht dokumentiert. Benötigt werden diese Informationen zur Fortführung des unterbrochenen Prozesses nach einer Wiederherstellung der zulässigen Betriebsfunktionen.

3 Virtuelle Maschine

Bei manchen Anwendungen ist es vorteilhaft, den Prozessor als virtuelle Maschine zu betreiben oder virtuelle Ein-/Ausgabe-Vorgänge durchzuführen. Der Mikroprozessor MC68010 unterstützt beide Konzepte. Damit dieser als virtuelle Maschine laufen kann, muß der Prozessor eine geschützte Umgebung besitzen. Beim Typ MC68000 war das mit der Unterscheidung in die privilegierten Ebenen für Benutzer und Supervisor möglich. Zusätzlich muß jede Operation, die auf die Supervisor- oder Steuerinformation zugreift, ausschließlich für den Supervisor privilegiert sein.

Beim Mikroprozessor MC68010 sind die Supervisor-Ressourcen vollständig vor Zugriffen vom Benutzer geschützt. Auf diese Weise kann das Betriebssystem der für den Benutzer privilegierten Ebene ablaufen, wobei durch die Traps gegen Privilegienverletzung derselbe Effekt hervorgerufen wird, als würde das Betriebssystem auf der Supervisorebene ablaufen. Damit kann der MC68010 als virtuelle Maschine benutzt werden oder virtuelle E/A-Vorgänge verarbeiten.

Mit dem Mikroprozessor MC68010 kann ein übergeordnetes Betriebssystem Operationen auf der Supervisorebene ausführen und die Verwaltung einer Anzahl anderer Betriebssysteme steuern, die auf der Benutzerebene arbeiten. Jedes Betriebssystem kann auf der Supervisorebene Verwendung finden. Wenn eine privilegierte Instruktion auf der Benutzerebene ausgeführt wird, prüft das übergeordnete Betriebssystem diese nach, führt sie bedingt aus und gibt sie an das ursprüngliche Betriebssystem zurück. Virtuelle E/A-Vorgänge ergeben sich direkt aus der Einrichtung zur Bus-Fehlerkorrektur.

4 Vektortabelle

Wenn ein Exception-Vorgang abläuft, entnimmt der Mikroprozessor MC68010 aus einer Vektortabelle mit 1024 Byte einen neuen Inhalt für den Programmzähler (PC). Nach einem Reset steht diese Tabelle auf dem Wert $00000000. Der MC68010 verfügt über Einrichtungen zur einfachen Relocation dieser Tabelle.

Im Prozessor wird ein 32-Bit-Vektor-Basisregister (VBR) initialisiert, indem es auf 0 gesetzt wird. Damit ist Kompatibilität mit dem Prozessor MC68000 gewährleistet. Geladen oder gelesen wird der Inhalt mit einer speziellen privilegierten Instruktion (MOVEC). Jedesmal, wenn der MC68010 einen Vektor für Exeption-Handling anfordert, wird zuerst ein Vektor-Offset durch Multiplikation einer Vektor-Nummer (die entweder intern erzeugt oder extern gelesen wird) mit 4 erzeugt. Danach wird dieser Vektor-Offset zum Inhalt des VB-Registers geladen, um die logische Adresse, von der der Vektor geholt wird, zu erzeugen.

Mit dem neuen VB-Register und der darin enthaltenen Einrichtung zur Vektor-Erzeugung verfügt der Program-

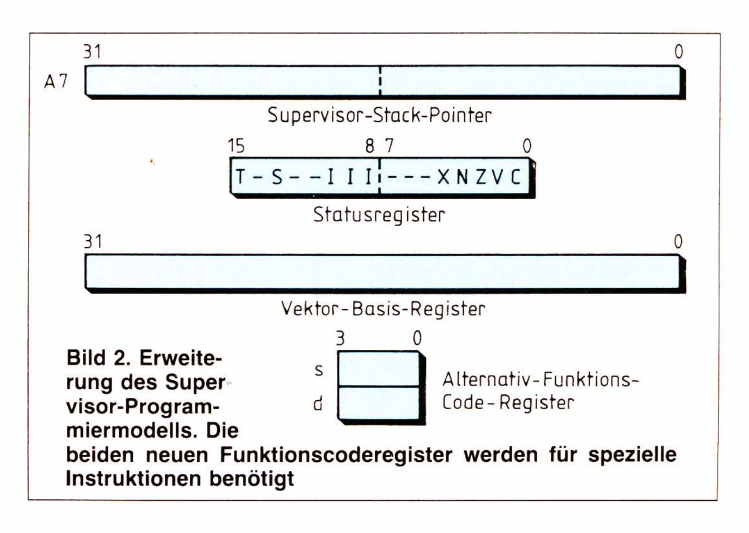


Bild 2. Erweiterung des Supervisor-Programmiermodells. Die beiden neuen Funktionscoderegister werden für spezielle Instruktionen benötigt

mierer über eine breitere Auswahlmöglichkeit für die Plazierung dieser Vektoren. Beim Einschalten des Systems oder einem Reset-Vorgang wird der anfängliche Stand des Programmzählers automatisch vom Mikroprozessor unter der logischen Adresse $00000004 geholt. Während des Betriebes kann das Betriebssystem die Vektoren für Exception-Routinen laden und diese im RAM-Bereich ablegen. Der Originalbereich ist $00000000...$000003FF. Um sie zu verschieben, lädt das Betriebssystem das VB-Register mit dem neuen Wert, der auf das untere Ende des gewünschten Adreßbereiches zeigt. Die Vektoren werden dann nacheinander aus dieser neu aufgestellten Tabelle entnommen. Damit ist es möglich, den Prozessor-Reset-Vektor sowie alle Initialisier-Interrupt- und Trap-Vektoren in einem ROM mit der Startadresse $00000000 abzulegen, aber nach der Programminitialisierung auf einen RAM-Bereich überzugehen, um verschiedene Vektoren für unterschiedliche Routinen, Betriebssysteme usw. vorsehen zu können.

5 Exception-Profil

Neben den Verbesserungen in bezug auf Bus-Fehlerkorrektur wurde beim MC68010 auch die Behandlung aller Exceptions allgemein verbessert. In diesem Prozessor für virtuelle Speicher werden neben dem Inhalt von Programmzähler und Statusregister auch noch die Nummer des benutzten Vektors im Supervisor-Stack abgelegt. Ein Handler könnte z. B. die Interrupts von einer Anzahl ADLCs MC6854 bedienen, aber nur mit einer allgemeinen Treiberroutine arbeiten, wobei die abgelegte Vektor-Nummer zur Bestimmung des spezifischen Bausteins benutzt wird. Auf diese Weise muß nur eine ADLC-Service-Routine geschrieben werden, um eine größere Zahl von Kommunikationsverbindungen zu unterstützen.

6 Verbesserte Software-Operationen

Gegenüber dem MC68000 ist der Befehlssatz des MC68010 in einigen Punkten verbessert. Es wurden nicht nur zusätzliche Instruktionen aufgenommen, sondern auch Ausführungszeiten verkürzt.

So arbeiten z. B. die Instruktionen „Clear" (CLR) und bedingtes Setzen (Scc) sowie die Operation MOVE SR/CCR nach EA schneller. Die Ausführungszeit von 32-Bit-Arithmetik- und Logik-Operationen wurde um zwei Taktzyklen verkürzt, was eine Verbesserung gegenüber dem MC68000 von maximal 25 % ausmacht.

Beim MC68010 wurde eine Einrichtung vorgesehen, die es dem ausführenden Betriebssystem erlaubt, Daten in einen oder aus einem Adreßbereich außerhalb des Supervisor-Datenbereichs zu transferieren. Dies führen die neuen MOVE-Befehle unter Verwendung der zusätzlichen Funktions-Code-Register aus. Die privilegierte Instruktion „MOVES" (zum Transfer in andere Adreßbereiche) ermöglicht es dem Betriebssystem, Daten von oder zu einer effektiven Adresse (EA) zu transferieren, während es den Funktionscode dazu bestimmt, der Ausgabewert während des Speicherzugriffes zu sein. Der Funktionscode wird durch das Quellen- oder Ziel-Funktionscode-Register festgelegt.

Ausführung einer MOVES-Instruktion auf der Supervisor-Ebene kann z. B. ein Daten-Langwort (2 Befehlsworte) von einem festgelegten Platz im Benutzerprogrammbereich holen und dieses in eines der Datenregister auf dem Chip laden. Danach kann das Betriebssystem die Instruktionen überprüfen. Die MOVES-Operation läßt den Supervisoroperanden auf einen Aufruf vom Benutzer aufnehmen sowie Daten von und zum Systempuffer zwischen zwei verschiedenen Benutzern hin und her verschieben.

Auf das Funktionscode-Register wird vom Betriebssystem mit einer weiteren zusätzlichen MOVE-Instruktion zugegriffen, die die Bezeichnung „MOVEC" trägt (Move Control Register).

Die MOVEC-Instruktion wird auch beim Schreiben oder Lesen des Vektor-Basisregisters benutzt. Das erlaubt dem Betriebssystem, den Bereich der Exception-Vektortabelle auf jeden Speicherplatz zu verschieben.

Viele Programme für den MC68000 machen starken Gebrauch von kurzen Programmschleifen mit der Instruktion DBcc (z. B. bei Kettenoperationen). Wenn die Schleife aus einer Ein-Wort-Instruktion besteht, die dem Befehl DBcc vorangeht, kann in einen speziellen Schleifen-Modus eingetreten werden. Wenn der MC68010 die beiden Instruktionen aufgenommen hat, durchläuft er die Schleife und transferiert Daten. Bis zum Verlassen der Schleife werden keine zusätzlichen Instruktionen aufgenommen. Der Mikroprozessor MC68010 verhält sich in diesem Fall so, als würde er eine große Zahl von Hochgeschwindigkeits-String-Befehlen zur Verfügung stellen.

7 „Relaxed Bus Timing"

Eine wichtige Veränderung in bezug auf das Zeitverhalten von Fehlern hebt beim MC68010 die Beschränkungen, die für viele heutige Systeme bei der Benutzung von Speichern bestehen, auf. Beim MC68010 ist es trotz Akzeptierens eines DTACK-Signals eine halbe Taktperiode später möglich, daß ein BERR-Eingangssignal DRACK zurückstellt und ein Bus-Fehler-Trap aktiviert. Dem Fehlererkennungssystem steht die halbe Taktperiode zur Verfügung, um zu entscheiden, ob gültige oder ungültige Daten am Prozessor anliegen. Damit sind in einem fehlererkennenden System Speicherbausteine zu verwenden, die nicht schneller sein müssen als in Systemen ohne Fehlererkennung.

Nach Unterlagen der Fa. Motorola, bearbeitet von *P. von Bechen*

Literatur

[1] *Daniels, G., Lösel, M.:* Ein 16-Bit-Mikroprozessor in HMOS-Technik. ELEKTRONIK 1979, H. 10, S. 47...53.

[2] *v. Bechen, P.:* 8-Bit-Prozessor bietet 32-Bit-Architektur. ELEKTRONIK 1983, H. 1, S. 43...46.

[3] *v. Bechen, P.:* 32-Bit-Prozessor erweitert 68000-Familie. ELEKTRONIK 1983, H. 6, S. 62...64.

Dipl-Ing. (FH) Werner Hilf

MC68020: 32-Bit-Prozessor für zukunftssichere Systemkonzepte

Noch vor wenigen Jahren hätte man als Mikroprozessoranwender an der Marktakzeptanz einer 32-Bit-CPU gezweifelt. Durch die innovativen Sprünge der Technik müssen aber auch Computer an den neuesten Stand angepaßt werden. Man denke dabei an Robotersteuerungen (digitale Regelungen) oder computerunterstützte Entwicklung (CAD/CAM), um nur zwei Beispiele zu nennen. Hierbei handelt es sich immer um Anwendungen, bei denen es auf Geschwindigkeit und höchste Genauigkeit (Auflösung) ankommt. Bisher konnten diese komplexen Aufgaben nur mit (Mini- bzw. Maxi-)Computern oder mit aufwendigen und empfindlichen Multiprozessorsystemen realisiert werden. Mit dem 32-Bit-Prozessor MC68020 steht ein Baustein zur Verfügung, der Leistungsmerkmale von Großcomputern aufweist, verbunden mit der klaren Hard- und Softwarestruktur der M68000-Familie. Hieraus ergibt sich ein weiteres Anwendungsgebiet: Als Zentraleinheit in leistungsfähigen Multiuser-/Multitasksystemen mit umfangreichen Schutzmechanismen. Mit Sicherheit stellt diese CPU eine ernsthafte Konkurrenz für Bit-Slice-Computer dar.

Während die Mikroprozessoren in Home- und Small-Business-Computern längst Einzug als Zentraleinheit gehalten haben, werden sie in Großcomputern bisher „nur" zur Peripheriesteuerung verwendet. Der Grund liegt darin, daß die wenigsten Prozessoren für diesen Anwendungsfall geschaffen sind. Eine Anwendung in diesen Systemen verlangt z. B. bestimmte Schutzvorkehrungen (verschiedene Modi), Verarbeitung von virtuellem Speicher, einfaches Task-(Context-)Switching, hohe Geschwindigkeit, großen Adressierbereich, ausgereiften Befehlssatz und effiziente Adressierungsarten; letztere vor allen Dingen zur optimalen Unterstützung von höher orientierten Sprachen (z. B. C, Pascal). Der erste Schritt in diese Richtung war die CPU MC68010. Zusammen mit dem MC68000 bildet sie daher die Grundlage für den Typ MC68020. In [1] wurde diese CPU bereits vorgestellt, hier sollen weitere Eigenschaften erläutert und Anwendungen gezeigt werden.

Eine wesentliche Änderung erfuhr die „Innenarchitektur" des MC68020 und der Zwei-Stufen-Mikrocode. Während beim MC68000/008/010 die ALU noch eine 16-Bit-Organisation besitzt [2], sind die einzelnen Blöcke im MC68020 (*Bild 1*) ausnahmslos für 32 Bit ausgelegt. Das hat nun zur Folge, daß die Blöcke, gesteuert durch den Mikrocode, untereinander parallel arbeiten können. So ist es nun möglich, im PC-Block eine Adreßrechnung vorzunehmen, während in den anderen Blöcken Daten hereingeholt und/oder Operanden berechnet werden. Dies trägt nicht nur zur Leistungssteigerung bei, sondern erlaubt darüber hinaus eine Auslegung aller möglichen Adressierungsarten (vergl. Abschnitt Adressierungsarten) bis 32 Bit.

Die Leistungsmerkmale auf einen Blick:
- alle Einrichtungen und Funktionen des MC68000/10
- Standardtaktfrequenz 16,67 MHz
- dynamische Busstruktur, 8-, 16- oder 32-Bit-Transfers
- wesentlich kürzere Bus- und Befehlszyklen
- komplette 32-Bit-Struktur
- 32-Bit-Adreß- und Datenleitungen (ohne Multiplex)

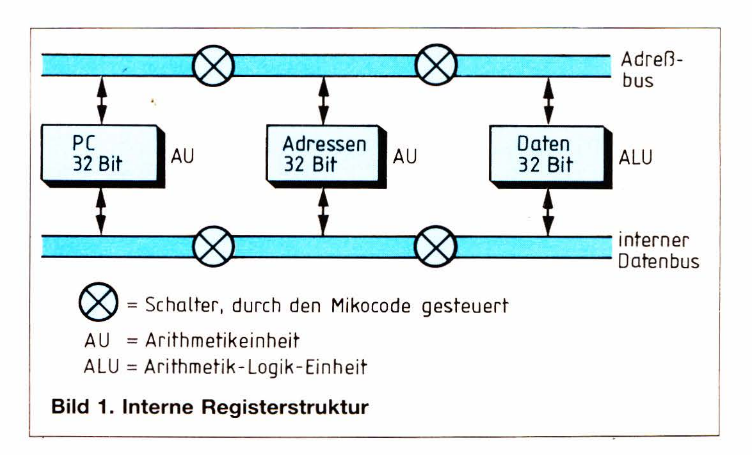


Bild 1. Interne Registerstruktur

Tabelle 1. Mit M- und S-Bit maßgebende Stackpointer

S	M	maßgebender Stackpointer		
0	0	User-Stackpointer	(USP)	wie 68000
0	1	User-Stackpointer	(USP)	
1	0	Interrupt (System) Stackpointer	(ISP)	SSP bei 68000
1	1	Master Stackpointer	(MSP)	

Tabelle 2. Trace-Funktionen

T1	T0	Trace-Funktionen (Einzelschritt)
0	0	kein Trace (wie 68000)
0	1	Trace-Exception nur bei Branch, Jump, JSR oder Return
1	0	Trace-Exception nach jedem Befehl (wie 68000)
1	1	nicht definiert, reserviert

– erweiterter Registersatz
– erweiterter, aufwärtskompatibler Befehlssatz (quell- und objektcode-kompatibel)
– erweiterte, aufwärtskompatible Adressierungsarten
– weitere Datentypen (gepackte BCD-Zahlen, Bit-Felder)
– Hochgeschwindigkeits-Befehlscache
– Coprozessor-Unterstützung.

1 Registersatz und Hardware

Die „Arbeits"-Register des MC68020 sind ausnahmslos 32 Bit breit, der Programmzähler macht keine Ausnahme. Unter dem Gesichtspunkt der sogenannten Aufwärtskompatibilität wurde dieses bereits bewährte Register-Grundmodell beibehalten (*Bild 2a*). Außer den vom MC68010 her bekannten „Vector-Base-Register" (VBR) und den „Alternate-Function-Code-Register" (SFC, DFC) erhielt der MC68020 zwei Supervisor-Stackpointer, die Interrupt- bzw. Masterstackpointer genannt werden (*Bild 2b*). Welcher Stackpointer benützt wird, entscheidet das ebenfalls neu implementierte M-Bit im Statusregister zusammen mit dem bekannten S-Bit (Bild 2a). Die möglichen Bitkombinationen und deren Bedeutung sind in *Tabelle 1* zusammengestellt.

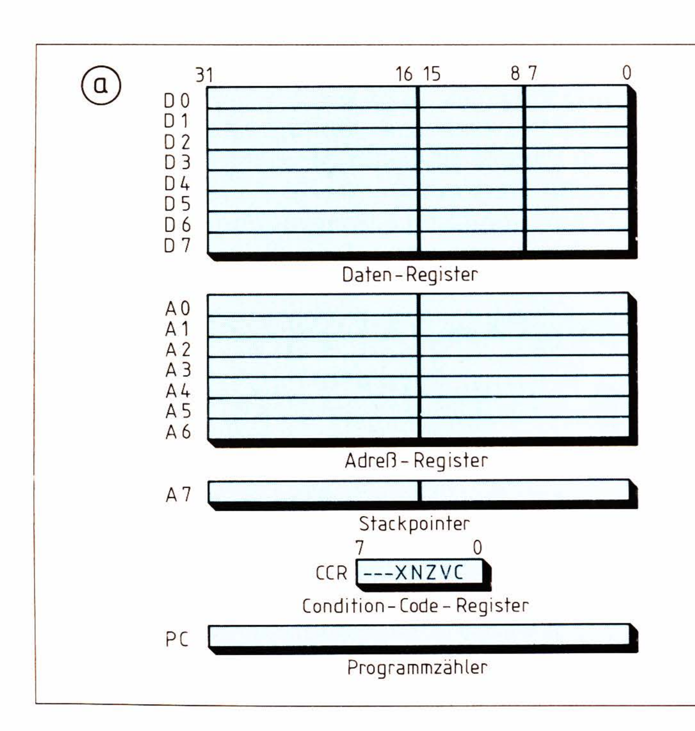
Der Vorteil zweier Supervisor-Stacks liegt z. B. bei der Task-Umschaltung. Dabei zeigt der MSP dann auf einen Task-Control-Block (TCB), der ISP auf den Interrupt-Stack. Vor der Taskumschaltung wird das M-Bit gesetzt. Alle Exceptions, außer Interrupts, verwenden dann den MSP. Eine Taskumschaltung, hervorgerufen durch einen Timerinterrupt, verändert demzufolge den MSP nicht, da er über einen eigenen Pointer verfügt (ISP).

Eine weitere Unterstützung beim Austesten von Programmen bringt das zusätzliche Trace-Bit „T_0" im Statusregister (Bild 2b). Die Bitkombinationen und deren Bedeutung zeigt *Tabelle 2*.

Diese erweiterte Trace-Möglichkeit bedeutet, daß man, ohne Haltepunkte setzen zu müssen, sich beispielsweise durch Unterprogramme durcharbeiten kann. Zwei Register, das „Cache-Steuer-Register" (CACR) und das Cache-Adreß-Register (CAAR) beziehen sich, wie der Name sagt, auf Cache-Manipulationen.

1.1 Signale des MC68020

Bussysteme ohne Multiplex, wie sie beim MC68020 zu finden sind, haben einen entscheidenden Geschwindigkeitsvorteil. Dieser Vorteil überwiegt bei weitem den Nachteil der benötigten Mehrzahl von Leitungen. Ein Mehraufwand an eventuell notwendigen Bustreibern muß dafür in Kauf genommen werden.



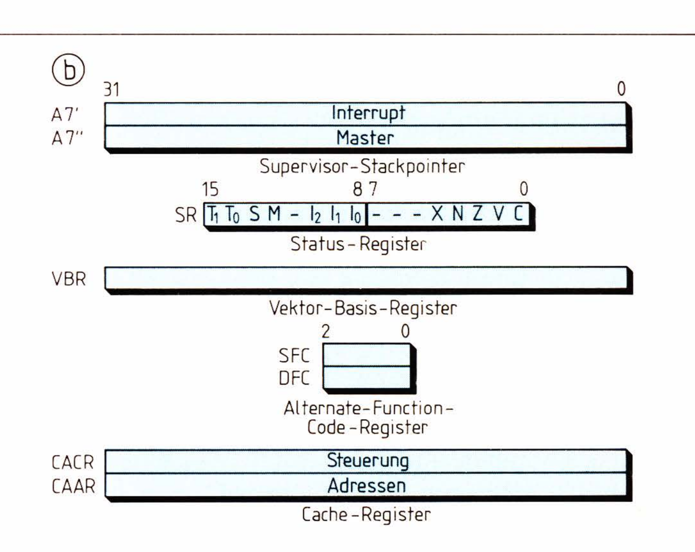


Bild 2. Programmiermodelle: a) Benutzer-Programmiermodell der Typen MC68000, 68008, 68010 und 68020; b) Supervisor-Programmiermodell des MC68020

Um die Vielzahl der Leitungen unterzubringen, wird der MC68020 in einem 13×13-Pin-Grid-Array geliefert. Neu eingeführte Signale sind mit ∗ gekennzeichnet. Auf diese Signale geht der folgende Abschnitt ein.

Die dynamischen Bussteuersignale setzen neue Maßstäbe auf dem Mikroprozessor-Sektor. Denn mit Hilfe der Signale DSACK0 und DSACK1 läßt sich hardwaremäßig bestimmen, ob ein 8-, 16- oder 32-Bit-Datentransfer durchgeführt werden soll, oder ob Wartezyklen benötigt werden, wobei die Portbreite innerhalb jedes Buszyklus geändert werden kann (dynamische Busstruktur).

Darüber hinaus kann dynamisch während eines Buszyklus die Datenbreite geändert werden. Damit die externe Hardware in einem derartigen Fall weiß, wieviel Bytes übertragen worden sind bzw. noch zu übertragen sind, wird die Anzahl über die SIZ0/1-Leitungen angegeben. Dadurch kann sich ein MC68020 seiner Hardwareumgebung dynamisch anpassen.

Ein Beispiel: MOVE.L D0,$20000

Dem Anwender stehen hier mehrere Möglichkeiten der Datenausgabe zur Verfügung. Der MC68020 führt zunächst einen Buszyklus aus und gibt die Daten auf den 32-Bit-Datenbus. Mit den DSACK-Signalen bestimmt der Anwender die Portbreite und damit die Anzahl der Buszyklen.

Fall 1:
DSACK0=0, DSACK1=0
dann wird dieser Befehl mit einem Buszyklus beendet. Der Inhalt des Datenregisters wird auf dem Datenbus D0...D31 ausgegeben.

Fall 2:
DSACK0=1, DSACK1=0
dieser Befehl wird mit zwei Buszyklen beendet, wenn in beiden Buszyklen mit dieser DSACK-Kombination geantwortet wird (16-Bit-Port). Den Inhalt des 32-Bit-Datenregisters gibt der Datenbus in zwei Worten D16...D31 aus.

Fall 3:
DSACK0=0, DSACK1=1
dieser Befehl wird mit vier Buszyklen beendet, wenn in allen vier Buszyklen mit dieser DSACK-Kombination geantwortet wird (8-Bit-Port). Das 32-Bit-Datenregister wird in vier Byte auf dem Datenbus D24...D31 ausgegeben.

Andere Kombinationen sind möglich, z. B. im Fall 2 zunächst mit 16-Bit-Portbreite zu antworten und darauffolgend zweimal mit einem Byte. Dann würde der Befehl in drei Buszyklen ablaufen, wobei ein Wort über den Datenbus D16...D31 und die nächstfolgenden beiden Byte über den Datenbus D24...D31 ausgegeben werden.

Allgemein erhält man daher die Portstruktur nach *Bild 4a*. Im Gegensatz zum MC68000/010 lassen sich Daten auf jede beliebige Adresse (auch ungerade) ablegen. Die Adreßleitungen A0, A1 und die Size-Leitungen SIZE0, SIZE1 bestimmen den Ort (Byte-Adresse) bzw. die Größe der Daten. Beispiele für ein 16-Bit- und 32-Bit-Port mit verschiedenen Datengrößen und Anfangsadressen zeigt *Bild 4b*.

Weitere Signale und deren Bedeutung:

RMC Zeigt einen laufenden Read-Modify-Write-Zyklus (RMW) an, bei den Befehlen TAS und CAS (wird nicht mehr über AS angezeigt)

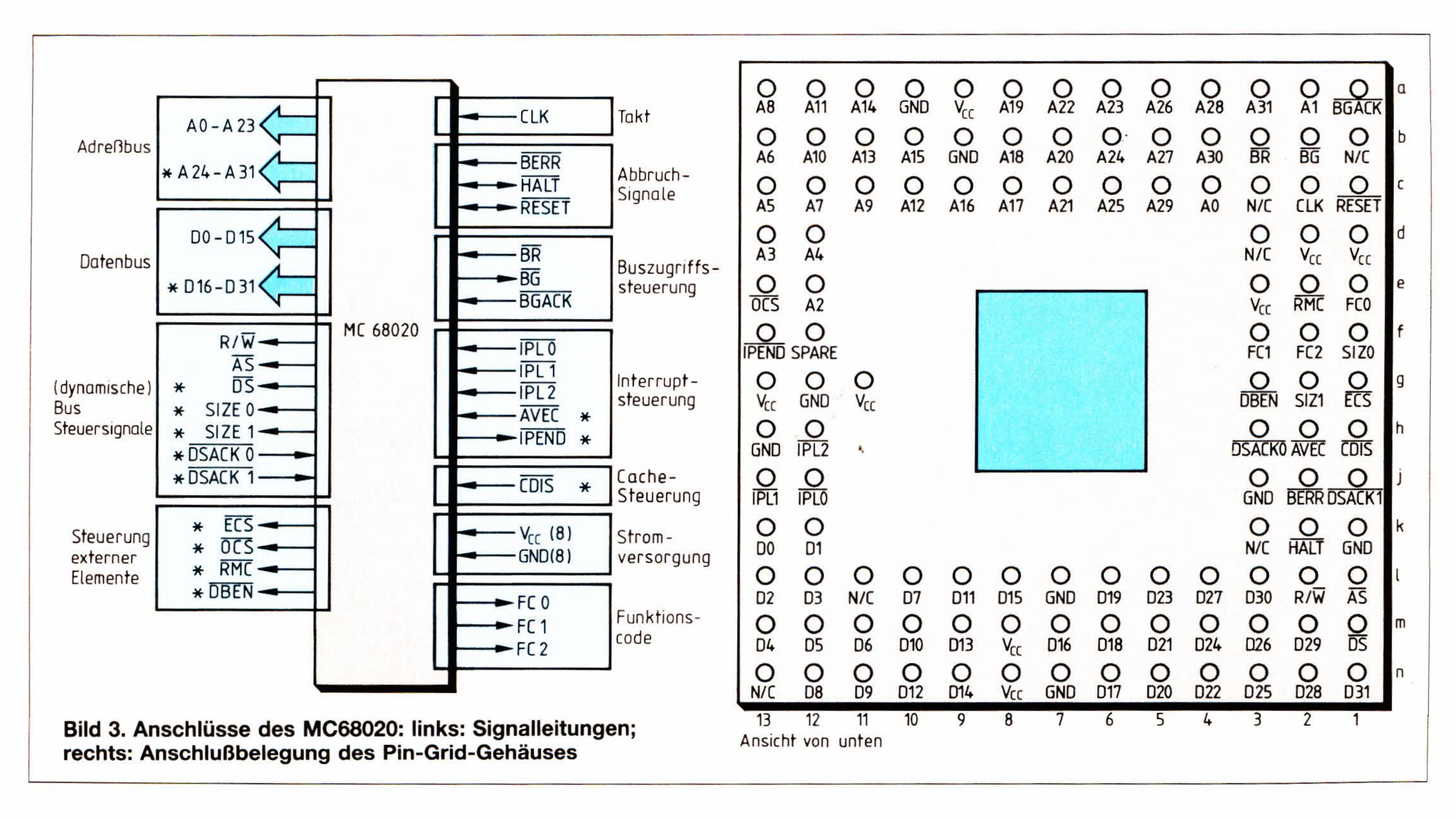


Bild 3. Anschlüsse des MC68020: links: Signalleitungen; rechts: Anschlußbelegung des Pin-Grid-Gehäuses

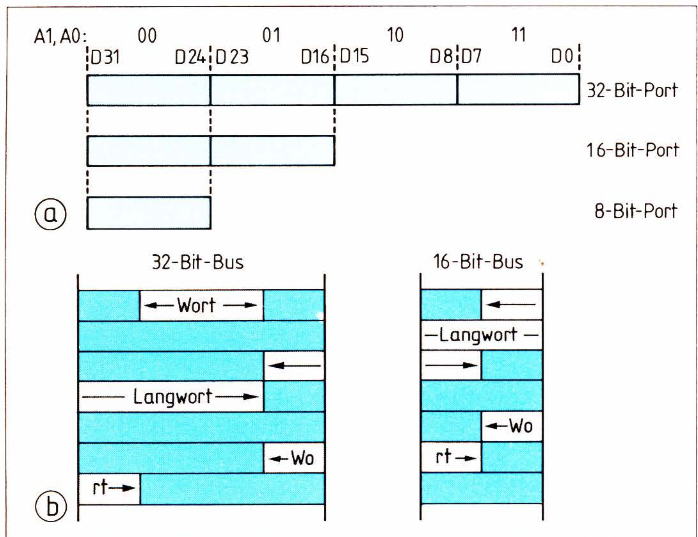


Bild 4. Daten-E/A: a) Datenbus-Portstruktur, b) verschiedene Datenzugriffe

Dn	d_{16}(PC)
An	d_8(PC, Xn)
(An)	ABSOLUTE SHORT
(An)+	ABSOLUTE LONG
−(An)	#IMMEDIATE
d_{16}(An)	#QUICK IMMEDIATE
d_8(An, Xn)	IMPLIED REGISTER

Bild 5. Adressierungsarten

Register-indirekt:			
(d, An)	(Xn∗scl)	(d, Xn∗scl)	(An, Xn∗scl)
(d, An, Xn∗scl)			
Speicher-indirekt:			
([d])	([d], d)	([An])	([d, An])
([An], d)	([d, An], d)	([Xn∗scl])	
Speicher-indirekt, Pre-Index:			
([d, Xn∗scl])	([Xn∗scl], d)	([d, Xn∗scl], d)	([An, Xn∗scl])
([d, An, Xn∗scl])	([An, Xn∗scl], d)	([d, An, Xn∗scl], d)	
Speicher-indirekt, Post-Index:			
([d], Xn∗scl)	([d], d, Xn∗scl)	([An], Xn∗scl)	([d, An], Xn∗scl)
([An], d, Xn∗scl)	([d, An], d, Xn∗scl)		

d ist ein Displacement von 8, 16 oder 32 Bits
Xn ist ein Indexregister, das jedes beliebige Register (Daten oder Adresse) sein kann (16 oder 32 Bit)
scl ist ein Skalierungsfaktor von 1, 2, 4 oder 8

Programmzähler relativ			
(d, PC)	(Xn∗scl)	(d, Xn∗scl)	(PC, Xn∗scl)
(d, PC, Xn∗scl)			
Speicher-indirekt:			
([d])	([d], d)	([PC])	([d, PC])
([PC], d)	([d, PC], d)	([Xn∗scl])	
Speicher-indirekt, Pre-Index:			
([d, Xn∗scl])	([Xn∗scl], d)	([d, Xn∗scl]), d)	([PC, Xn∗scl])
([d, PC, Xn∗scl])	([PC, Xn∗scl], d)	([d, PC, Xn∗scl], d)	
Speicher-indirekt, Post-Index:			
([d], Xn∗scl)	([d], d, Xn∗scl)	([PC], Xn∗scl)	([d, PC], Xn∗scl)
([PC], d, Xn∗scl)	([d, PC], d, Xn∗scl)		

Bild 6. Zusätzliche Adressierungsarten des MC68020

$\overline{\text{AVEC}}$ Autovektor (Ersatz für VPA bei MC68000)
$\overline{\text{IPEND}}$ „Interrupt pending", kann verwendet werden, um die externe Hardware früher in Kenntnis zu setzen, daß ein Interrupt kommen wird.
$\overline{\text{ECS}}$ „External cycle start", wird aktiviert, wenn der Prozessor einen externen Buszyklus beginnt (früheste Erkennung). Kann zusammen mit $\overline{\text{OCS}}$ z. B. für externes Cache, dynamische RAMs usw. verwendet werden.
$\overline{\text{OCS}}$ „Operand cycle start", wie $\overline{\text{ECS}}$, jedoch nur bei Operandentransfer.
$\overline{\text{DBEN}}$ „Data buffer enable", zur Freigabe externer Treiber, um evtl. Buskonflikte zu vermeiden.

2 Adressierungsarten

Den Adressierungsarten wurde bei der Entwicklung besondere Aufmerksamkeit geschenkt. Fand man beim MC68000 noch gewisse Einschränkungen bei Adreßdistanzen vor (nur 8- oder 16-Bit-Adreßdistanz möglich), so stehen dem Programmierer jetzt 4 GByte zur Verfügung. Überhaupt ist der MC68020 außerordentlich vielseitig in bezug auf die Adressierungsarten, denn viele wurden zusätzlich implementiert. Um dem Leser selbst einen Vergleich zu ermöglichen, sind in *Bild 5* in Kurzform die Adressierungsarten des MC68000/010 und in *Bild 6a* und *b* die zusätzlichen des MC68020 dargestellt.

Bild 6a zeigt einen Überblick der Register-indirekten bzw. Speicher-indirekten Adressierungsarten. Durch die in der Einführung beschriebenen internen Strukturen sind alle diese Adressierungsarten sowohl programmzählerrelativ (Bild 6b) einsetzbar als auch ausnahmslos mit 32 Bit Adreßdistanzen (Displacements) möglich. Der Programmierer kann damit zwischen insgesamt 62 (!) Varianten wählen.

Was bedeuten nun die einzelnen Adressierungsarten und wie setzen sie sich zusammen? Eine sehr übersichtliche Aufschlüsselung gibt das Bild 7.

Hieraus ist zu entnehmen, daß die Adressierungsart aus einem Basisregister (entweder ein Adreßregister oder der Programmzähler) plus einem Displacement (0, 16 oder 32 Bit) plus einem Index (Daten- oder Adreßregister, 16 oder 32 Bit ohne oder mit Skalierungsfaktor 1, 2, 4 oder 8) plus einem weiteren Displacement (0, 16 oder 32 Bit) besteht. Der Index kann selbst eine indirekte Adresse sein und kann vor oder nach der „Indirektion" dazu gezählt werden. Im übrigen ist insgesamt eine doppelt indirekte Adressierung möglich (*Bild 8*). Mit dem Skalierungsfaktor hat der Anwender die Möglichkeit, in Tabellen einzugreifen, ohne den Indexpointer zu zerstören. Viele dieser Adressierungsarten finden Einsatz in Compilern, die sehr häufig auf Tabellen (Arrays oder Pointer) zugreifen; aber auch in Betriebssystemen oder bei Peripheriebausteinen (z. B. Disk-Controllern), bei denen über eine Tabelle weitere Datenbereiche (Baumstruktur) angesprochen werden. Ein Beispiel nur soll hier herausgegriffen werden: Die Addition eines

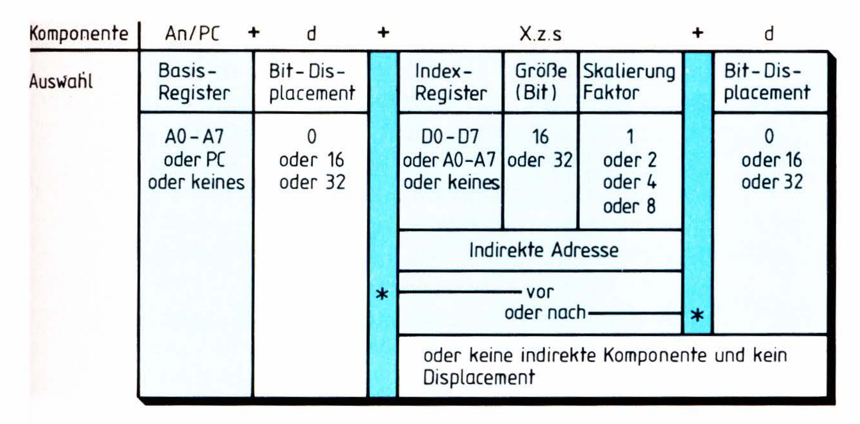


Bild 7. Zusammensetzung der Adressierungsarten

Tabelle 3. Bedeutung der DSACK-Signale

$\overline{DSACK1}$	$\overline{DSACK0}$	Ergebnis
1	1	Wartezyklen
1	0	8-Bit-Port
0	1	16-Bit-Port
0	0	32-Bit-Port

Tabelle 4. Bedeutung der SIZE-Signale

SIZE1	SIZE0	Ergebnis
0	0	Langwort
0	1	Byte
1	0	Wort
1	1	3 Byte

Wertes, der über mehrere Tabellen zu finden ist, mit dem Datenregister D5:
ADD ([BASE,A1,D3*4],DIST),D5
Es handelt sich hierbei um eine der Adressierungsarten Speicher-indirekt mit Pre-Index. Wie der Quelloperand zu finden ist, zeigt Bild 8.

Obwohl viele Adressierungsarten beim MC68020 hinzugekommen sind, findet man hier Aufwärtskompatibilität auch auf der Opcodeebene. Denn Änderungen wurden nur in den unbenützten Bits des Erweiterungswortes bei der Adressierungsart „Adreßregister-indirekt mit Index plus Adreßdistanz" des MC68000 [2] vorgenommen.

3 Befehlssatz

Nicht nur neue Befehle wurden implementiert, sondern vom MC68000 übernommene Befehle hinsichtlich Operand oder Speicherbereich erweitert (*Tabelle* 5).

Der Anwender hat nunmehr bei der Multiplikation und Division (mit oder ohne Vorzeichen) mehr Möglichkeiten wie beim MC68000, so ist folgende Arithmetik erlaubt:

MULU.W/MULS.W — 16*16 → 32 Bit (wie MC68000)
MULU.T/MULS.L — 32*32 → 32 Bit
MULU.L/MULS.L <ea>, Di : Dj — 32*32 → 64 Bit
DIVU.W/DIVS.W — 32/16 → 16-Bit-Ergebnis und 16-Bit-Rest (wie MC68000)
TDIVU/TDIVS — 32/32 → 32-Bit-Ergebnis und 32-Bit-Rest
DIVU.L/DIVS.L <ea>, Di : Dj — 64/32 → 32-Bit-Ergebnis und 32-Bit-Rest

Eine Gegenüberstellung von Ausführungszeiten für Multiplikation bzw. Division bei MC68020/MC68010 zeigt *Tabelle 6*.

Eine Reihe von neuen Befehlen, die z. T. die zusätzlichen Register, den Befehls-Cache-Speicher oder die Coprozessoren unterstützen oder mit denen die Bit-Feldmanipulationen und die BCD-Umrechnungen vollzogen werden, bieten dem Programmierer eine höhere Leistungsfähigkeit und eine größere Vielfalt. Im folgenden sind die neu zum MC68000/68010-Befehlssatz hinzugekommenen Instruktionen aufgelistet:

BFEXTU/BFEXTS — Bit Field Extract (signed/unsigned)
BFINS — Bit Field Insert
BFFFO — Bit Field Find First One
BFTST — Bit Field Test
BFCLR — Bit Field Clear
BFSET — Bit Field Set
BFCHG — Bit Field Change
BKPT #n — Breakpoint
CHK2 — Check Register against Bounds

Tabelle 5. Erweiterungen von Befehlen des MC68020

	zusätzliche Adressierungsarten	32-Bit-Operation
TEST		×
CMPI	×	×
CHECK	×	×
LINK		×
BRA, Bcc		×
EXT von Byte		×
MUL		×
DIV		×

Tabelle 6. Gegenüberstellung MUL/DIV von 68020/68010

	MC68020 (max.)		MC68010 (max.)	
	16,67 MHz	Taktzyklen	12,5 MHz	Taktzyklen
MULU 16	1,44	24	3,20	40
MULU 32	2,64	44	–	–
MULS 16	1,44	24	3,36	42
MULS 32	2,64	44	–	–
DIVU 16	2,76	46	8,64	108
DIVU 32	5,04	84	–	–
DIVS 16	3,36	56	9,76	122
DIVS 32	5,76	96	–	–

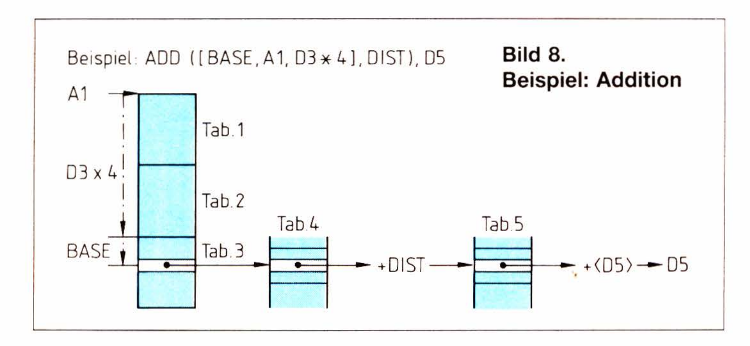


Bild 8. Beispiel: Addition

CMP2	Compare Register with Bounds
Tcc	Trap on Condition
TPcc #xxx	Trap on Condition
PACK	Pack BCD
UNPK	Unpack BCD
CALLM	Call Module
RTM	Return Module
CAS	Compare and Swap
Coprozessor-Befehle*	
cpBcc	Branch on Coprocessor Condition
cpDBcc	Test Coprocessor Condition, Decrement and Branch
cpGen	General Coprocessor Operation
cpRESTORE	Coprocessor Restore
cpSave	Coprocessor Save
cpScc	Set Conditionally
cpTRAPcc	Trap on Coprocessor Condition

* diese Befehle sind allgemeingültig für Coprozessoren, für den Fließkomma-Coprozessor werden eigene Mnemonics verwendet (z. B. FMUL, FDIV, usw.), die sich auf diese Befehle beziehen.

Einige Erläuterungen zu den neuen Befehlen:
Die bedingten Abfragen wurden dahingehend erweitert, daß sie auch direkt zu Exceptions führen können (Tcc). Mit dem CHK-Befehl konnte bislang geprüft werden, ob man sich zwischen 0 und einer effektiven Adresse befand. Mit dem CHK2-Befehl ist eine Abfrage zwischen zwei effektiven Adressen möglich. Ein integrierter „Barrel-Shifter" sorgt nun dafür, daß alle Schiebe- und Rotationsbefehle mit einer konstanten Geschwindigkeit ablaufen, unabhängig von deren Anzahl.

Die bewährten Bit-Manipulationen der MC68000-Familie wurden erweitert mit den Bit-Feld-Manipulationen. Es können Bit-Felder mit oder ohne Offset gelesen, geändert, getestet, gesetzt, eingefügt und zurückgeschrieben werden. Außerdem ist ein Suchen nach der ersten „Eins" in einem Bit-Feld möglich. Der Offset kann im Bereich von $\pm 2^{31}$ liegen, das Feld umfaßt Bit 1...32 und kann Langwortgrenzen überschreiten. Beispiele zeigen die *Bilder 9a* und *9b*.

Bild 9a ist ein allgemeines Beispiel, *Bild 9b* zeigt die Vorgehensweise beim Verarbeiten von Records, bei der wiederum komplexe Adressierungsarten Verwendung finden. Es wird ein Bit-Feld in ein Datenregister (D0) eingelesen. Das Bit-Feld steht innerhalb eines Records. Ein weiteres Datenregister (D3) zeigt auf das Bild-Feld innerhalb des Records. Ein Adreßregister (A3) zeigt auf die Records. Zusammen mit einem Displacement (REC5) wird das gewünschte Record (REC5) lokalisiert. Das Bit-Feld umfaßt jedoch nur 24 Bit und wird mit einem Offset n ausgelesen, der in einem Datenregister (D4) steht. Zu weiteren Anwendungsgebieten gehört die Grafikverarbeitung, mit diesen Befehlen lassen sich auf einfache Weise bestimmte Pixel ansprechen.

Eine Weiterführung des TAS-Befehles ist der CAS. Er führt auch einen unteilbaren Zyklus durch (RMW) und ist daher für Multiprozessorsysteme geeignet. Ein Beispiel:
CAS Dn,Dm,⟨ea⟩
Er testet nicht nur ein Bit, sondern vergleicht den Inhalt einer effektiven Adresse <ea> mit einem Datenregister (Dn). Sind die beiden Inhalte identisch, wird das Datenregister Dm in ⟨ea⟩ geschrieben.

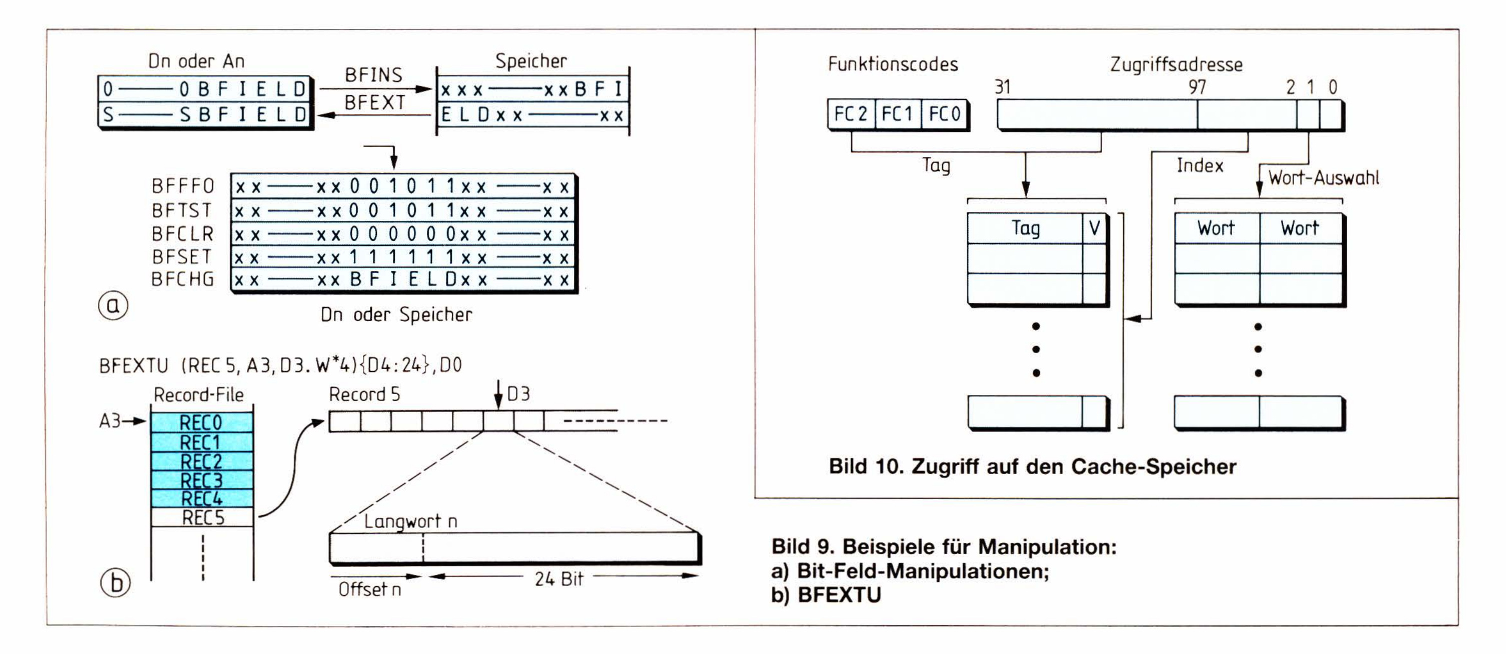


Bild 10. Zugriff auf den Cache-Speicher

Bild 9. Beispiele für Manipulation:
a) Bit-Feld-Manipulationen;
b) BFEXTU

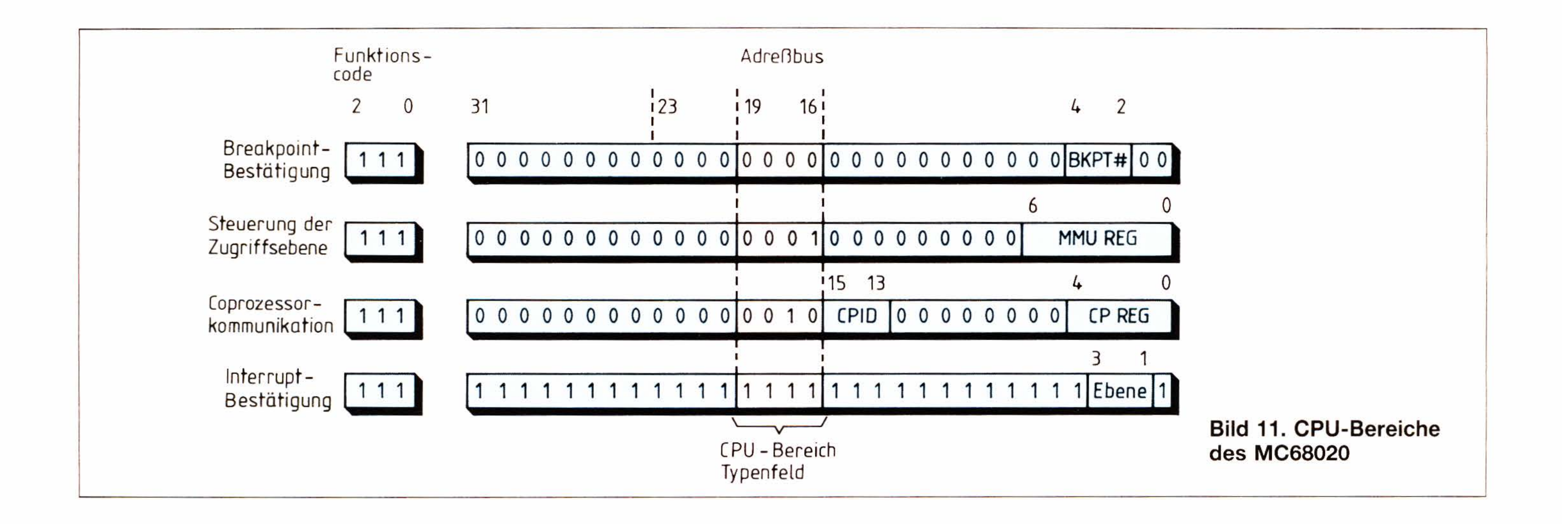


Bild 11. CPU-Bereiche des MC68020

Der Modulmechanismus (CALLM, RTM) des MC68020 stellt eine Erweiterung des Supervisor-User-Schutzes dar und erlaubt den Aufruf eines Modules mit bestimmten Zugriffsrechten. Bis zu 256 Privilegebenen sind möglich (MC68000 : 2). Ein Modul ist ein Stack-Bereich (Frame), der mit CALLM kreiert wird. In diesem Stack-Bereich stehen Informationen wie z. B. Zugriffsrecht (Ebene), Datenpointer, Adreßpointer usw. Unterstützt wird dieses Konzept hauptsächlich im Zusammenspiel mit der externen Hardware, z. B. der geplanten „Paged Memory Management Unit" (PMMU). Der Prozessor gibt spezielle Informationen über die Funktionscodeleitungen und Adreßbus aus („CPU-Space", *Bild 11*).

4 Cache-Speicher

Der MC68020 besitzt auf dem Chip einen Cache-Speicher, der immer dann beschrieben wird, wenn der Mikroprozessor nach außen zugreift, um sich einen Opcode zu holen oder einen Operanden (Erweiterungswort), der dem Opcode folgt. Dann führt der MC68020 einen Mehrfach-Wort-Befehls-Prefetch aus. Durch einen komplizierten internen Hardwaremechanismus wird während eines Programmablaufes überprüft, ob der im externen Speicher liegende Befehl bereits in das Cache hineingeholt wurde und dort vorliegt. Ist dies der Fall, muß keine Busaktivität seitens des Mikroprozessors erfolgen, es sei denn, der auszuführende Befehl greift seinerseits auf eine Speicherzelle zu. Dennoch wird durch diese Eigenschaft der Bus wesentlich entlastet, was bei Multiprozessorsystemen zu größerem Durchsatz verhilft. Dort bildet der Bus bekanntlich das „Nadelöhr". Weitere Vorteile des Cache zeichnen sich ab: So könnten ganze Programmteile, z. B. Abfrageschleifen, die nur interne Register betreffen, ohne jeden weiteren Buszugriff ablaufen.

Die Cache-Kapazität des MC68020 beträgt 256 Byte. Das heißt, bei einer Cache-Wortgröße von 32 Bit sind 64 Einträge möglich (Bild 10). Cache-Manipulationen stehen dem Anwender ebenfalls mit Hilfe zweier Register zur Verfügung (Bild 2b). So kann der Anwender das Cache bei Bedarf ausschalten (Debug- und Emulationsphasen) oder den gesamten Cache löschen. Interessant ist auch die Fähigkeit, den Cache „einzufrieren". Anwendungsbeispiele können Interrupt- oder Unterprogramme sein, die, nachdem sie im Cache stehen, eingefroren werden, d. h. gespeichert bleiben. Jedesmal, wenn dieses Programm aufgerufen wird, ist kein erneutes Holen notwendig. Diese Programme können mit interner Geschwindigkeit ablaufen. Eine weitere programmierbare Eigenschaft besteht darin, eine Adresse innerhalb des Cache zu überspringen (mit CAAR), z. B. Breakpoints.

5 Coprozessor-Schnittstelle

Eine weitere Neuerung ist die integrierte Coprozessor-Schnittstelle. Hier handelt es sich, wie man vermuten könnte, keinesfalls um eine Hardware-, sondern um eine Softwareschnittstelle. Der Coprozessor MC68881 benötigt keine speziellen Leitungen. Vielmehr erfolgt der Anschluß über die bereits vom MC68000 her bekannten Adreß-, Daten- und Funktionscode-Leitungen. Darin (und in der dynamischen Busstruktur des MC68881) liegt im übrigen das Geheimnis, daß der Floating-Point-Coprozessor MC68881 auch an den MC68008, MC68000 und MC68010 anzuschließen ist.

Wie ist demnach die Softwareschnittstelle beschaffen? Betrachtet man zunächst *Bild 11* und ruft man sich den Zustand der Adreß- und Funktionscodeleitungen beim MC68000 im Falle eines Interrupterkennungszyklus in Erinnerung, ergibt sich folgendes Bild:

Sämtliche Adreßleitungen sind in diesem Zustand auf „1". Die Adreßleitungen A1...A3 geben die Interruptprioritätsebene (IPL) wieder, deren Interrupt gerade angenommen wurde. Die Funktionscodeleitungen sind hierbei ebenfalls alle „1". Beim MC68020 wurde dieser Bereich, d. h. wenn FC0-2 logisch „1" ist, erweitert. Die Adreßleitungen A16...A19 definieren den Typ. So erhält man je eine andere Kombination, wenn der CALLM-

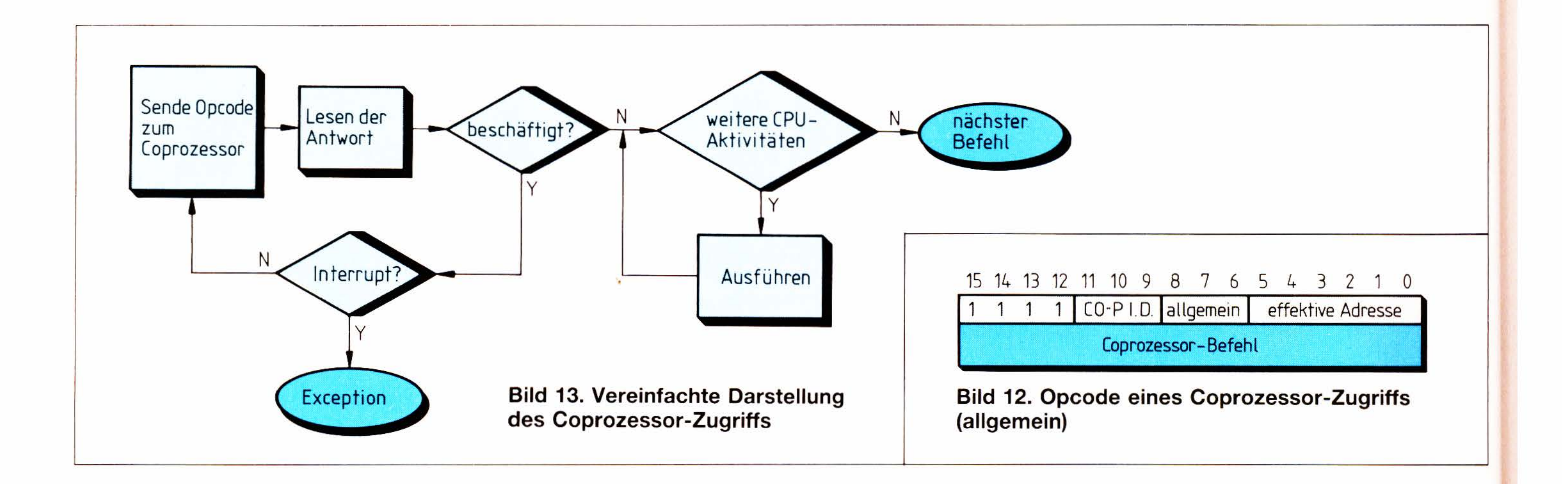


Bild 13. Vereinfachte Darstellung des Coprozessor-Zugriffs

Bild 12. Opcode eines Coprozessor-Zugriffs (allgemein)

Befehl (Access-Level Control) oder der BKPT-Befehl (Breakpoint Acknowledge) aufgerufen wird.

Jeder Coprozessorbefehl führt dazu, daß die Leitungen FC0-2 beim MC68020 ebenfalls auf „1" gesetzt werden. Die Unterscheidung für einen externen Baustein, ob es sich um einen Interruptzyklus oder Coprozessorzugriff handelt, liegt nun in den Adreßleitungen A5...A15 und A20...A31, die logisch „0" sind. A17 ist „1", A13...A15 geben die sog. Coprozessoridentität an, die im Befehlssatz verankert ist. Dies könnte hardwaremäßig decodiert werden. Damit sind in einem System bis zu acht Coprozessoren anschließbar. Es ist denkbar, z. B. zwei mit der Berechnung eines transzendentalen Ausdrucks zu beauftragen, die dann nahezu parallel zur Verfügung stehen. Die Adreßleitungen A0...A4 bestimmen die Registerauswahl im Coprozessor.

Grundsätzlich führt jeder Coprozessorbefehl dazu, daß die höchsten vier Bit des Opcode (Bit 12...15) „1" gesetzt werden. In den Bits 9...11 steckt die bereits beschriebene Coprozessor-Identität, die Bits 0...5 geben die effektive Adresse an (*Bild 12*).

Bit 6...8 unterscheidet die Coprozessorzugriffe. Diese sind:
- General (alle mathematischen Funktionen)
- Branch on Condition
- Trap on Condition
- Set on Condition
- Decrement and Branch on Condition
- Save (Retten der Register)
- Restore (Rückspeichern der Register)

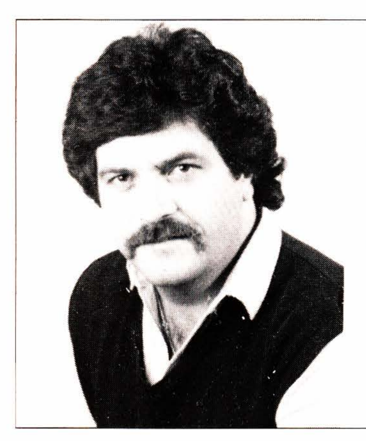
In den nachfolgenden Worten befinden sich, je nach Befehl die Erweiterungsworte, Adreßdistanzen oder zusätzliche Befehlsformate.

Der Coprozessor erkennt demzufolge anhand der Funktionscodeleitungen und der Coprozessorinstruktion (werden mit dem Chip-Select des Coprozessors verknüpft), daß eine Information vorliegt und holt sich die weiteren Daten, die benötigt werden (können auch im Speicher liegen). Der Coprozessor bestimmt selbständig den Ablauf und die Synchronisation mit der CPU, er ist sogar in der Lage, Traps einzuleiten.

Dem MC68020 fallen bei dieser Schnittstelle die Aufgaben zu, die Operanden zu laden und zu speichern, die effektive Adresse zu errechnen und die Traps zu verarbeiten. Eine vereinfachte Darstellung des MC68020 Coprozessor-Zugriffs zeigt *Bild 13*.

6 Zusammenfassung

Mit dem MC68020 wird dem Anwender eine CPU angeboten, die Mikroprozessormärkte erschließt, die bislang Mini- oder sogar Großcomputern vorbehalten waren. Darüber hinaus lassen sich neue Anwendungen mit diesem Prozessor realisieren. Man denke hierbei an Applikationen, die bislang mangels Geschwindigkeit oder aus Platzgründen nicht zu verwirklichen waren; weitere Gründe sind Preisrelationen oder zu hohe Verlustleistung.



Werner Hilf ist in Schopfheim/Krs. Lörrach geboren. Er hat die Elektronik „von der Pike auf" gelernt. 1967 Lehre in einem Entwicklungs- und Forschungslabor. Studium der Nachrichtentechnik 1970 an der Fachhochschule Karlsruhe. Zunächst Analog-Elektronik, insbesondere Prozeß- und Regeltechnik. Mit dem Siegeszug des Mikroprozessors und fasziniert von der Digitaltechnik begann er mit vielfältigsten Aufgaben aus dem Bereich der Hard- und Software mit diversen Mikroprozessoren. Seit 1979 ist er Leiter der Mikroprozessor-schulung bei Motorola in München. Zahlreiche Vorträge im Inland, europäischen Ausland und Afrika. Sein Wissen vermittelt er seit 1980 auch als Lehrbeauftragter an der Fachhochschule München und der Technischen Akademie in Esslingen.

Literatur

[1] *von Bechen, P.*: 32-Bit-Prozessor erweitert 68000-Familie. ELEKTRONIK 1983, H. 6, S. 62...64.

[2] *Hilf, W., Nausch, A.*: M68000-Familie, Teil 1: Grundlagen und Architektur. tewi-Verlag, München.

[3] *Hilf, W., Nausch, A.*: M68000-Familie, Teil 2: Anwendung und 68000-Bausteine. tewi-Verlag, München.

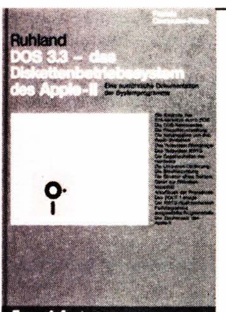
[4] Motorola, Inc. MC68020 Präsentation, interne Veröffentlichung.

[5] Motorola, Inc. MC68020 User's Manual, Mai 1984.

Der sichere Einstieg in die EDV



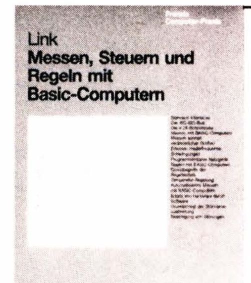
Franzis-Verlag, München



DOS 3.3 – das Diskettenbetriebssystem des Apple II

Eine ausführliche Dokumentation der Systemprogramme. Von B. **Ruhland**. 255 S., 12 Abb., DM 48,–. ISBN 3-7723-7691-6

Der Anwender kann für jede Aktion des Betriebssystems die treffende Programmstelle finden und verstehen.



Messen, Steuern und Regeln mit Basic

Programmierung und lauffähige Programme für die elektronische Meßtechnik. Von W. **Link**. 141 S., 143 Abb., DM 38,–. ISBN 3-7723-7411-5

Mit diesem Buch entdecken Computerfans neue Anwendungsgebiete für ihre Basic-Maschine.

Hardware-Erweiterungen für den ZX 81

Der nachträgliche Einbau von Schnittstellen zum Messen, Steuern und Regeln. Von O. **Merker**. 199 S., 132 Abb., DM 48,–. ISBN 3-7723-7811-0

Der interessierte Leser findet mit diesem Buch genügend Anregungen, seine eigenen Problemlösungen zu entwickeln.



Der sichere Einstieg in Pascal

Der leichte Weg zum selbständigen Programmieren in Pascal. Von R. **Busch**. 188 S., 46 Abb., DM 48,–. ISBN 3-7723-7861-7

Sinn und Zweck dieses Buches ist es, dem Lernenden einen bequemen Einstieg in diese moderne Programmiersprache zu schaffen.
Systematisch erklärt der Autor zunächst die wichtigsten Sprachelemente. Diese werden anhand von praktischen Beispielen eingeübt und laufend wiederholt. Ein Grundwortschatz in Pascal entsteht.
Und was kann der Leser und Benutzer dieses Buches am Ende? Das exakte Definieren von Algorithmen hat er gelernt. Er beherrscht einen Basisumfang von Pascal. Grammatik und Syntax sind geläufig. Er hat die unbändige Lust auf mehr und der Karriere als Pascal-Programmierer steht nichts mehr im Wege.

Ganz gleich, ob es um Betriebssysteme, Hardware-Erweiterungen, Steuerungsaufgaben oder um das eigentliche Programmieren geht. Der verantwortliche Planer und Anwender von EDV in Wirtschaft, Industrie oder Labor findet unter diesen Büchern seine kompetenten Grundlagen, handfeste Anleitungen und Hilfen. Die bringen ihn echt weiter.

Dieses Wissen richtig angewandt, schafft es, den vielen anderen mindestens um die berühmte Nasenlänge voraus zu sein.



Mikrocomputer-Technik praxisnah

Eine verständliche Einführung in die Technik und Arbeitsweise für alle, die es genau wissen wollen. Von D. **Benda**. 189 S., 96 Abb., DM 44,–. ISBN 3-7723-7841-2

Der Autor hat den Wissensstoff in Text und Bild bewußt leichtverständlich und stets anwendungsbezogen dargestellt und aufbereitet. Die Lektionen sind anschaulich, regen zum systematischen Arbeiten an und entsprechen den modernen Erfordernissen des Lernenden. Besonders hervorzuheben sind die Abschnitte zum Üben und Selbsttesten. Sie erlauben ein jederzeitiges Überprüfen des eigenen Wissensstandes. Wer diesen Band systematisch durchgeht und seine Aufgaben und Übungen auch wirklich erledigt, der besitzt ein tiefgreifendes Wissen über die Mikrocomputertechnik. Er kann jetzt die Funktionsabläufe der Mikrocomputer nachvollziehen und in Betrieb nehmen.

Mit Computern steuern

Aufbau und Anwendung von Einplatinen-Mikrocomputern. Von H. **Feichtinger**. 178 S., 144 Abb., DM 38,–. ISBN 3-7723-7221-X

Mit Einplatinencomputern steuern und regeln, ab wann sich das lohnt und wie das zu machen ist, darum geht es in diesem Buch. Der Autor zeigt, daß der technische Aufwand dafür gering ist und behandelt selbstverständlich nur Typen, die auf dem Markt sind und wohl auch länger bleiben werden.



Betriebssystem CP/M

Vom Monitorprogramm zum Mehrbenutzersystem. Von J. **Plate**. 351 S., mit 30 Abb., DM 56,–. ISBN 3-7723-7521-9

Geboten wird die wohl benutzerfreundlichste und umfassendste Beschreibung des Betriebssystems CP/M. Der Leser wird vom einfachen Monitorprogramm über das weit verbreitete Betriebssystem CP/M zu den Multiuser- und Multitasking-Betriebssystemen geführt. Der Autor geht sehr in die Tiefe und ins Detail.



Datenfernübertragung per Computer

Der C 64 findet Kontakt zu Mailboxen und Datenbanken über einen Akustikkoppler. Von T. **Winzer**. Ca. 136 S., ca. 54 Abb., ca. DM 32,–. ISBN 3-7723-8071-9

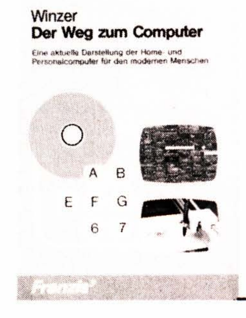
Der Autor stellt das umfangreiche Thema locker, verständlich dar. Er öffnet so dem Computer-Fan das Tor zu unendlich vielen, interessanten Daten. Er knüpft einen neuen Medienverbund, der preiswert und im wahrsten Sinne des Wortes handsam ist.



Was Sie über Software wissen sollten

Die Handhabung der Software in Wirtschaft und Industrie. Von F. **Haugg**. 215 S., 107 Abb., DM 38,–. ISBN 3-7723-7721-1

Doch mit diesem Buch sind Sie ein kompetenter Gesprächspartner für alle, besonders für die EDV-Leute.



Der Weg zum Computer

Eine aktuelle Darstellung der Home- und Personalcomputer im Vorfeld der Kaufentscheidung. Von T. **Winzer**. 176 S., 70 Abb., DM 28,–. ISBN 3-7723-7731-9

Jedes Kapitel ist leicht zu lesen, wichtige Fachbegriffe werden zunächst definiert, bevor sie im Text Verwendung finden.



Der Heim-Computer 8085

Der einfache Nachbau eines Computers mit Programmieranleitungen von R. **Keil**/G. **Keil**. 174 S., 13 Abb., DM 38,–. ISBN 3-7723-7471-9

Allen Computerfreunden wird der Nachbau eines Heimcomputers leichtgemacht. Benötigt wird dazu der 8-Bit-Mikroprozessor 8085.

(82)

Was der HX-20 von Epson noch zusätzlich alles kann

Wer das wissen will, der greife zu diesem Band. Wo das Original-Handbuch aufhört, dort fangen die Autoren an.

Fünf Punkte sind es, die in die Augen stechen:

1. Zusätzliche Kenntnisse über Hardware und Basic-Interpreter (z. B. PEEK und POKE) bilden die Basis für zusätzliche Applikationen.
2. Über dreißig neue, sorgfältig erprobte Anwendungen aus der Praxis des Tages brauchen nur abgeschrieben werden. Sie laufen.
3. Hier öffnet sich das Tor zur Maschinensprache des HX-20, mit der bestimmte Programme eben wesentlich schneller sind (Assembler, Disassembler).
4. Interessante Hardwareerweiterungen machen den Hand-Held wesentlich leistungsfähiger (Speichererweiterung, Schnittstelle für Messen, Steuern und Regeln).
5. Für die routinierten Anwender ein Glossar mit Speicherbelegung und Systemroutinen.

Hard- und Software für den Epson HX-20



Heiße Applikationen für den attraktiven Akten-taschen-Computer. Von Jürgen Plate und Wolfram Lechner.

144 Seiten, 20 Abbildungen. Lwstr-gebunden. DM 38.–.
ISBN 3-7723-8041-7

Franzis'

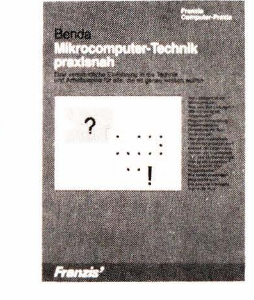
der große Fachverlag für angewandte Elektronik und Informatik
Franzis-Verlag, München

(101)

Zum Üben und Selbsttesten Wie intelligent ist ein Mikrocomputer?

Wie lautet der Untertitel: Eine verständliche Einführung in die Technik und Arbeitsweise für alle, die es genau wissen wollen. Und damit hat der Autor den Nagel auf den Kopf getroffen. Ganz klar, eindeutig und übersichtlich werden die Zusammenhänge von der Hardware- und Software und ihre gegenseitige Abhängigkeit vermittelt.
Der Leser lernt die Funktionsabläufe der Mikrocomputer nachzuvollziehen und die schaltungstechnischen Zusammenhänge von der CPU bis zur Schnittstelle zu begreifen. Weitere Erkenntnisse gelten der Softwarestruktur, die die schaltungstechnischen Funktionsabläufe (Hardware) in Betrieb setzen.
Wer diesen Band systematisch durchgeht und seine Aufgaben und Übungen auch wirklich erledigt, der besitzt ein tiefgreifendes Wissen über die Mikrocomputertechnik. Er kann jetzt die Funktionsabläufe der Mikrocomputer nachvollziehen und in Betrieb nehmen.

Mikrocomputer-Technik praxisnah



Eine verständliche Einführung in die Technik und Arbeitsweise für alle, die es genau wissen wollen. Von Dietmar Benda

189 Seiten, 96 Abbildungen, 22 Tabellen. Lwstr-geb. DM 44.–.
ISBN 3-7723-7841-2

Franzis'

der große Fachverlag für angewandte Elektronik und Informatik
Franzis-Verlag, München

(90)

Mehr Freude und Wissen durch den Selbstbau eines Heimcomputers

Wer hat mehr von einem Heimcomputer?
Der, der ihn sich für ein paar hundert Mark fertig gekauft hat, oder der, der ihn sich für fast das gleiche Geld selber gebaut hat?
Allen Computerfreunden wird nun mit Hilfe dieses Buches der Nachbau eines Heimcomputers leicht gemacht. Benötigt wird dazu der bewährte 8-Bit-Mikroprozessor 8085. Schritt für Schritt wird dem Anwender der Aufbau der Hardware auseinandergesetzt. Ausführlich erläutert werden die Bedienung und die Fehlersuche. Zahlreiche einfache Programmbeispiele lassen die praktischen Erfahrungen mit dem selbstgebauten Gerät schnell anwachsen. Über das Interface erarbeitet man sich auch noch das Entscheidende: Die Steuerungs- und Regelungstechnik.
Fazit: Mit Hilfe dieses Buches kann man sich gekonnt mit der Arbeitsweise von Mikrocomputersystemen auseinandersetzen.

Der Heim-Computer 8085



Der einfache Nachbau eines Computers mit Programmier-anleitungen.
Von Reinhard **Keil** und Günter **Keil.**

174 Seiten mit 13 Abbildungen und 14 Tabellen. Lwstr-geb. mit Schutzumschlag DM 38.–
ISBN 3-7723-7471-9

Franzis'

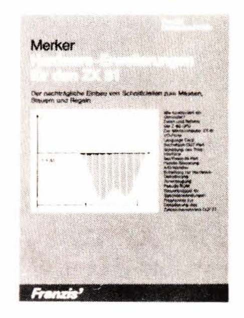
der große Fachverlag für angewandte Elektronik und Informatik
Franzis-Verlag, München



Der ZX-81 sinnvoll aufgerüstet für Meß-, Steuer- und Regelaufgaben

Der ZX 81 ist mehr als ein Programmier-Übungsgerät. Wer das noch nicht gewußt hat, erkennt das sofort, wenn er das Buch aufschlägt; wer das geahnt hat, erkennt sofort die Fülle der Möglichkeiten, den ZX 81 für neue, vielseitige Aufgaben zurechtzutrimmen.
Vorgestellt werden verschiedene Hardware-Zusätze. Mit diesen Baugruppen läßt sich der Computer für Meß-, Steuer- und Regelaufgaben sowie zur Tonerzeugung einsetzen. Eine Centronier-Schnittstelle zum Anschluß eines handelsüblichen Druckers sowie eine „languagecard“, mit der sich Änderungen im Betriebssystem durchführen lassen, werden ebenfalls beschrieben. In allen Fällen sind detaillierte Bauanleitungen mit Platinenvorlagen angegeben. So wird gleichzeitig die Funktion der einzelnen Schaltungen erklärt. Das ist ein wesentlicher Gesichtspunkt, denn er legt die Verknüpfung von Hard- und Softwareaspekten offen zu Tage.
Der kleine ZX 81, nach diesem Buch sinnvoll aufgerüstet, leistet plötzlich fast das Gleiche, wie ein wesentlich teuerer PC-Computer.

Hardware-Erweiterungen für den ZX 81



Der nachträgliche Einbau von Schnitt-stellen zum Messen, Steuern und Regeln. Von Oskar **Merker**.

220 Seiten, 112 Abbildungen, Lwstr-geb. DM 48.–
ISBN 3-7723-7811-0

Franzis'

der große Fachverlag für angewandte Elektronik und Informatik
Franzis-Verlag, München

(54)

Arun Kamat

„Fremde“ Interface-Bausteine an der CPU 68000

Systementwürfe mit CPUs der 68000-Familie können manchmal praktisch nicht realisiert werden, weil bestimmte Peripheriebausteine noch nicht zur Verfügung stehen. Mit wenig zusätzlichem Aufwand ist es allerdings möglich, Interface-Bausteine anderer Prozessorfamilien mit einer CPU aus der 68000-Familie zusammen zu betreiben und hohe Systemleistung zu erreichen. Der Beitrag zeigt das an folgenden Peripheriebausteinen: serieller Kommunikations-Controller AmZ8530 (SCC), FIFO-Ein/-Ausgabe-Baustein AmZ8038 (FIO), programmierbarer Intervall-Zeitgeber Am9513 (PIT), universeller DMA-Controller Am9516 (UDC), programmierbarer Interrupt-Controller Am9519 (PIC).

1 Serieller Kommunikations-Controller

1.1 *AmZ8530 ohne Interrupt*

Der serielle Kommunikations-Controller AmZ8530 ist die derzeit schnellste serielle E/A-Einheit auf dem Markt. Von diesem Baustein werden alle modernen Protokolle unterstützt. Eine große Zahl von Eigenschaften können vom Benutzer programmiert werden, wodurch sich sehr hohe Flexibilität beim Entwurf ergibt.

Beim Anschluß an die CPU 68000 (*Bild 1*) sind die Datenbusse direkt miteinander verbunden. Mit Hilfe des Adreß-Decodierers Am29806 werden die Adressen und das CS-Signal für die Peripherieeinheit erzeugt. Außerdem produziert der Baustein das Signal ANYE, das das Schieberegister 74LS164 zum Erzeugen von Wartezyklen benötigt. Das Schieberegister 74LS174 steuert die Zahl der Wartezyklen über den Eingang C, der wiederum das DTACK-Eingangssignal für die CPU steuert. Auf diese Weise ist es möglich, daß die Signale RD und WR die erforderliche Länge von 400 ns erreichen. A_1 erzeugt das C/D-Eingangssignal für den AmZ8530, während die übrigen Adreßleitungen mit dem Adreß-Komparator Am29809 verbunden sind.

Mit Hilfe der ODER-Gatter 74LS32S werden die Signale ODS, LDS und R/W in die AmZ8530-Signale RD und WR umgesetzt. Um den Zeitbezug zwischen CS und RD für den AmZ8530 einhalten zu können, verzögert das Flipflop 74LS74 die fallende Flanke von RD.

1.2 *AmZ8530 mit Interrupt*

Mit zusätzlicher externer Hardware kann die Interruptstruktur des AmZ8530 zur Verbesserung der Systemleistung herangezogen werden. Hier wird lediglich die Schaltung zum Erzeugen der entsprechenden Interrupt-Steuersignale beschrieben.

Der 3:8-Decodier-Baustein 74LS138 erhält die Prozessor-Statussignale FC_0, FC_1 sowie FC_2 und decodiert diese zum Interrupt-Acknowledge-Signal INTA des AmZ8530. A_1, A_2 sowie A_3 bestimmen die unterschiedlichen Interrupt-Ebenen, die vom 74LS138 durch Decodierung festgelegt werden. Es ist zu beachten, daß INTA das Signal RD des Prozessors sperrt.

Das Schieberegister 74LS164 verzögert das RD-Signal so lange, daß ausreichend Zeit zum Einschwingen der

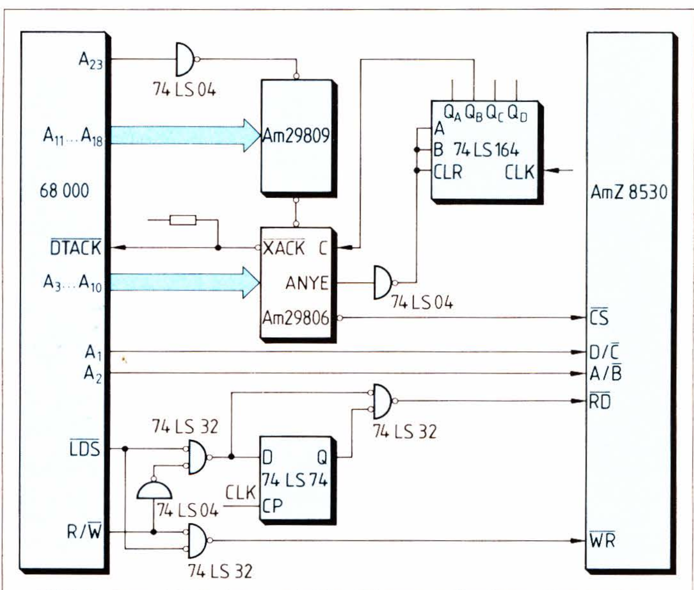


Bild 1. Anschluß des seriellen Kommunikations-Controllers AmZ8530 an die CPU 68000 ohne Interrupt-Mechanismus. Der SCC-Baustein ist das derzeit schnellste E/A-IC auf dem Markt

Interrupt-Daisy-Chain zur Verfügung steht. Darüber hinaus produziert dieses Schieberegister ein RD-Signal, das dafür sorgt, daß der Vektor früher auf dem Bus verfügbar ist, um das DTACK-Eingangssignal für die CPU 68000 zu erzeugen. Mit Hilfe des Bausteins 74LS148 werden die Interrupt-Request-Signale des AmZ8530 codiert und als gewünschte Interruptebene an den 68000 weitergeleitet.

1.3 AmZ8530 im Vergleich mit anderen Bausteinen

Der serielle Kommunikations-Controller AmZ8530 bietet in 68000-Systementwürfen z. B. gegenüber den bis jetzt benutzten Bausteinen MC6850, 6852 oder 6854 Vorteile. Von Motorola ist die Einführung eines leistungsfähigen seriellen E/A-Bausteins geplant. Bis dieser auf dem Markt erscheint, ist der AmZ8530 der einzige SCC-Baustein, der dem Benutzer die Möglichkeit zur Programmierung von Funktionen für hochwertige Anwendungen bietet (*Tabelle 1*). Für die Verwendung des AmZ8530 sprechen folgende Punkte:

- er unterstützt alle modernen Protokolle, wie HDLC, SDLC, IBM-Bisync
- mit dem AmZ8530 können Übertragungsraten von maximal 1,5 MBit/s erreicht werden
- im AmZ8530 sind Zusatzfunktionen untergebracht, über die vergleichbare Produkte nicht verfügen. Bei-

Tabelle 1. Vergleich der Kommunikations-Controller

Merkmal	AmZ 8530 SCC	Z-80 SIO	Z-80 DART	MC6850 AIA
Betriebsspannung	+5 V	+5 V	+5 V	+5 V
Gehäuse	40pol. DIP	40pol. DIP	40pol. DIP	24pol. DIP
Technologie	N-Kanal-MOS	N-Kanal-MOS	N-Kanal-MOS	N-Kanal-MOS
Taktsignale	ein	ein	ein	zwei
Kanäle	zwei	zwei	zwei	einer
Übertragungsrate	1,5 MBit/s	800 KBit/s	800 KBit/s	1 MBit/s
Minimale Taktzykluszeit	165 ns	250 ns	250 ns	500 ns
Fehlerschutz	CRC, CCITT	CRCC, CCITT	CRC, CCITT	Parity
Interrupt-Einrichtung	programmierbar	programmierbar	programmierbarer Vektor-Interrupt	nicht programmierbar
Baudratengenerator	auf dem Chip, programmierbar	nein	nein	nein
Digital-PLL	auf dem Chip	nein	nein	nein
Modem-Steuerung	Handshake-Interf.	Handshake-Interf.	Handshake-Interf.	Handshake-Interf.
Abbruch-Signale	programmierbar	nein	nein	nein
Protokolle	asynchron synchron Byte-orientiert IBM-BiSync Bit-orientiert HDLC, SDLC	asynchron	asynchron	asynchron
Auto-Echo	ja	nein	nein	nein
Lokale Schleife	ja, für lokale oder Fernwartung	nein	nein	nein

Merkmal	MC6852 SSD	MC6854 ADL	8251 PCI	8273 PPC
Betriebsspannung	+5 V	+5 V	+5 V	+5 V
Gehäuse	24pol. DIP	28pol. DIP	28pol. DIP	40pol. DIP
Technologie	N-Kanal-MOS	N-Kanal-MOS	N-Kanal-MOS	N-Kanal-MOS
Taktsignale	zwei	zwei	ein	ein
Kanäle	einer	einer	einer	einer
Übertragungsrate	?	?	64 KBit/s	64 KBit/s
Minimale Taktzykluszeit	280 ns	500 ns	250 ns	250 ns
Fehlerschutz	Parität	CRC	Parität	CRC
Interrupt-Einrichtung	nicht programmierbar	nicht programmierbar	nein	programmierbar
Baudratengenerator	nein	nein	nein	nein
Digital-PLL	nein	nein	nein	auf dem Chip
Modem-Steuerung	Handshake-Interf.	Handshake-Interf.	Handshake-Interf.	Handshake-Interf.
Abbruch-Signale	nein	programmierbar	nein	nein
Protokolle	synchron Bit-orientiert HDLC, SDLC	synchron Bit-orientiert HDLC, SDLC	asynchron synchron Byte-orientiert IBM, BiSync	asynchron synchron Bit-orientiert HDLC, SDLC
Auto-Echo	nein	nein	nein	nein
Lokale Schleife	nein	nein	nein	nein

spiel dafür ist der Baudraten-Generator. Ein externer Generator muß nicht vorgesehen werden, was zu einer Einsparung von etwa 20 DM/System führt. Die Baudrate ist programmierbar und kann an das Peripheriegerät angepaßt werden

- eine weitere Einsparung von 20 DM ergibt sich aus der auf dem Chip befindlichen digitalen PLL-Schaltung. Eine solche Funktion erfordert diskret aufgebaut mindestens 5...6 MSI- oder einen PLA-Baustein. Mit der DPLL-Schaltung ist Trennen der Daten vom Taktsignal bei selbsttaktender Codierung möglich
- Auto-Echo und lokale Schleife verbessern Möglichkeiten zur Diagnose durch einfache Überwachung der ausgesendeten Daten.

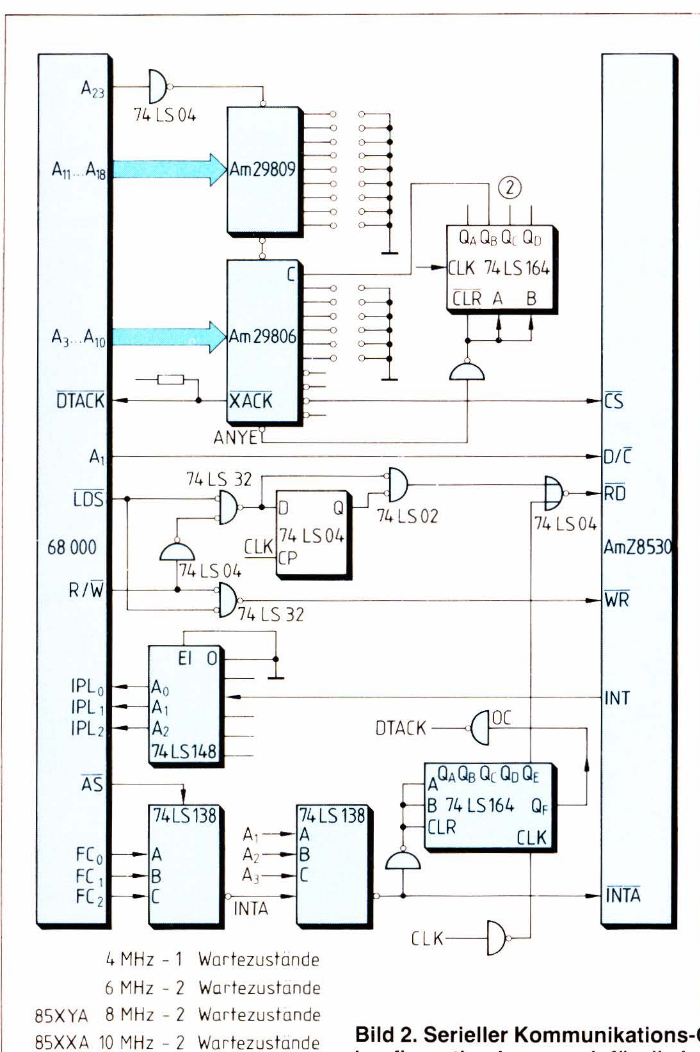
2 FIFO-Schaltungen

2.1 *FIO-Baustein AmZ8038 ohne Interrupt*

Der FIO-Baustein AmZ8038 läßt sich als Puffer zwischen zwei CPUs bei Multiprozessor-Anwendungen einsetzen, wobei sich die bidirektionale Transfer-Möglichkeit positiv bemerkbar macht. Im Beispiel nach *Bild 3* werden zwei FIO-Bausteine in einer Hochleistungsschaltung verwendet.

Die Datenbusse von 68000 und AmZ8038 sind direkt miteinander verbunden. Vom Adreßdecodierer Am29806 werden die 8-Bit-Adressen zur Verfügung gestellt. Außerdem erzeugt dieser Baustein das Signal ANYE, das als Eingangssignal für das Schieberegister 74LS164 dient. Da DTACK gesteuert wird, erzeugt dieses Register die gewünschte Anzahl von Wartezyklen. WR und RD haben damit die erforderliche Dauer von 400 ns. A_1 erzeugt das C/D-Eingangssignal für den AmZ8038, während die übrigen Adreßleitungen mit dem Adreßkomparator Am29809 verbunden sind. Die RD- und WR-Eingänge der beiden Bausteine AmZ8038 sind miteinander verbunden, wodurch effiziente Programmierung erreicht wird.

Mit Hilfe der ODER-Gatter 74LS32 werden die Signale DS, LDS sowie R/D in die AmZ8038-Signale RD und WR umgesetzt. Mit Hilfe des Flipflops 74LS74 wird die fallende Flanke von RD verzögert, um das zeitliche



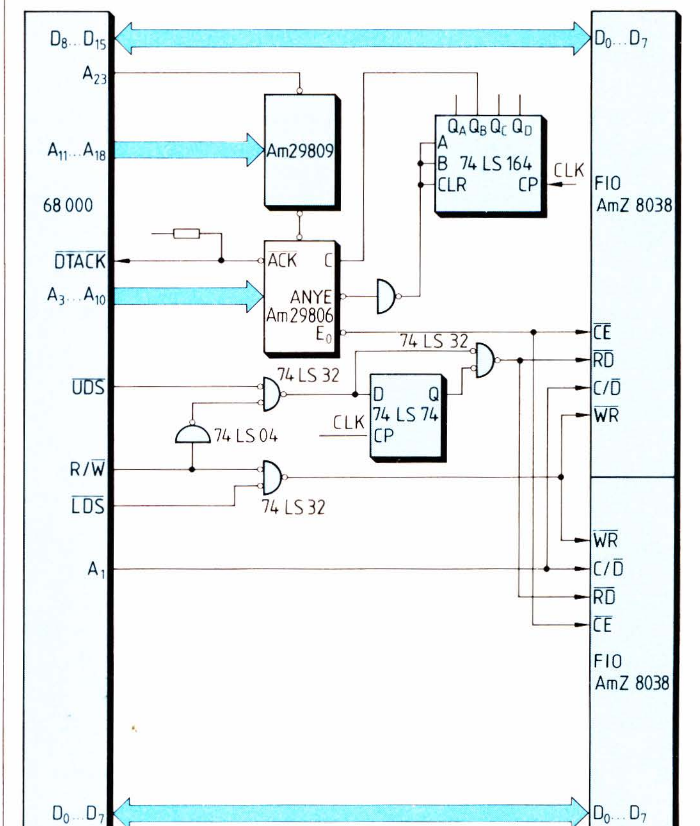


Bild 3. Als Daten-Puffer zwischen zwei oder mehreren CPUs in einem Multiprozessor-System verwendet man FIFO-Bausteine. In diesem Beispiel sind zwei FIO-ICs vom Typ AmZ8038 parallel geschaltet, um die Datenwortbreite zu erreichen

Bild 2. Serieller Kommunikations-Controller mit Interrupt-Mechanismus. Eine ähnliche Schaltungskonfiguration kann auch für die Interrupt-Steuerung der anderen Peripheriebausteine Verwendung finden

Verhältnis zwischen CS und RD entsprechend der Spezifikation des AmZ8038 einzuhalten.

Um die Interruptfunktion dieses Bausteins auszunutzen, kann eine ähnliche Schaltung, wie sie bereits für den AmZ8530 vorgestellt wurde, konfiguriert werden.

2.2 AmZ8038 im Vergleich mit anderen Bausteinen

Beim AmZ8038 handelt es sich um einen 128-Byte-FIO-Baustein, der über mehrere zusätzliche Merkmale verfügt, die hohe Flexibilität beim Schaltungsentwurf ermöglichen (*Tabelle 2*). Wichtige Merkmale des FIO-Bausteins AmZ8038 sind:

- bidirektionaler Datentransfer ist bei Multiprozessoranwendungen, bei denen zwei oder mehrere CPUs miteinander kommunizieren, besonders nützlich (derzeit einziger bidirektionaler FIFO-Baustein)
- das programmierbare Interface ermöglicht einfachen Anschluß an CPUs mit völlig verschiedenen Busstrukturen, ohne daß eine externe Logik erforderlich ist
- unbegrenzte Erweiterungsfähigkeit ermöglicht einfachen Ausbau existierender Systeme
- weitere wichtige Eigenschaften sind sieben Interruptquellen und ein 3-Draht-Handshake-Betrieb (IEE-488-kompatibel).

3 Intervall-Zeitgeber

3.1 Programmierbarer Intervall-Zeitgeber Am9513

Beim programmierbaren Intervall-Zeitgeber Am9513 (PIT) handelt es sich um einen Hochleistungs-Intervall-Timer, der sich leicht an die CPU 68000 anschließen läßt (*Bild 4*). Die Datenbusse der CPU 68000 und des PIT Am9513 sind direkt miteinander verbunden. Mit Hilfe des Adreßdecodierers Am29806 werden die 8-Bit-Adressen und das CS-Signal gewonnen. A_1 findet als C/D-Eingangssignal für den Am9513 Verwendung, während die restlichen Adressen mit dem Adreßkomparator 29809 verbunden sind.

Mit Hilfe der ODER-Gatter werden die Signale LDS und R/W in die Am9513-Eingangssignale RD und WR umgewandelt. Das Flipflop 74LS74 verzögert die fallende Flanke von RD, um den Zeitbezug zwischen CS und RD, der für den Baustein Am9513 spezifiziert ist, einzuhalten.

3.2 Am9513 im Vergleich mit anderen Bausteinen

Der programmierbare Intervall-Zeitgeber Am9513 ist ein Controller der dritten Generation für Hochleistungssysteme. Ein Anschluß an die CPU 68000 erfordert nur wenig externe Logik. Von Motorola ist der Baustein

Tabelle 2. Vergleich der FIFO-Bausteine

Merkmal	AmZ8038	AmZ8060	Am2812	Am2813
Betriebsspannung	+5 V	+5 V	+5 V, –12 V	+5 V, –12 V
Gehäuse	40pol. DIP	28pol. DIP	28pol. DIP	28pol. DIP
Technologie	N-Kanal-MOS	N-Kanal-MOS	N-Kanal-MOS	N-Kanal-MOS
Kapazität	128 x 8	128 x 8	32 x 8	32 x 9
Datenübertragungsrate	1 MHz	1 MHz	1 MHz	1 MHz
Durchtaktungszeit	250 ns	250 ns	10 µs	10 µs
Seriell/Parallel-Betrieb	parallel	parallel	seriell/parallel	seriell/parallel
Bidirektional	ja	ja	nein	nein
Mustererkennung	ja	ja	nein	nein
Interrupt-Struktur	Vektor-/vektorfreie Interrupts	nein	nein	nein
Handshake-Signale	interlocked	interlocked	nein	nein
Interface-Einrichtungen	programmierbar für Z-Bus- und andere CPUs	nur Z-Bus	nein	nein

Merkmal	Am2841	Fairchild 9043	Fairchild 9423	Western Digital 1502
Betriebsspannung	+5 V, –12 V	+5 V	+5 V	+5 V, –12 V
Gehäuse	16pol. DIP	24pol. DIP	24pol. DIP	28pol. DIP
Technologie	N-Kanal-MOS	bipolar	bipolar	N-Kanal-MOS
Kapazität	64 x 4	16 x 4	64 x 4	40 x 9
Datenübertragungsrate	1...2 MHz	10 MHz	10 MHz	0,5 MHz
Durchtaktungszeit	10 µs	450 ns	2,5 µs	?
Seriell/Parallel-Betrieb	parallel	seriell/parallel	seriell/parallel	parallel
Bidirektional	nein	nein	nein	nein
Mustererkennung	nein	nein	nein	nein
Interrupt-Struktur	nein	nein	nein	nein
Handshake-Signal	nein	nein	nein	nein
Interface-Einrichtungen	nein	nein	nein	nein

MC68230 verfügbar, der einen Zähler, parallele E/A-Anschlüsse und einen Interrupt-Controller auf einem Chip vereinigt. Im Gegensatz dazu bietet der PIT Am9513 folgende Vorteile:

- verfügt in einem 40poligen Gehäuse über fünf Zeitgeber (68230: ein Timer/48 Anschlüsse)
- bietet fünfmal soviel verschiedene Betriebsarten wie der 68230 und kann daher an unterschiedlichste Erfordernisse des Systems angepaßt werden
- Auf- und Abwärts-Zählbetrieb mit Zählerstandspeicherung sowie Möglichkeiten zur Frequenzteilung bietet der MC68230 nicht
- der Zähler des Am9513 ermöglicht auch den Betrieb als Uhr, wodurch bei Echtzeit-Anwendungen mehrere MSI-Schaltungen überflüssig werden.

Wenn es der Entwurf erforderlich macht, daß die drei Funktionen in einem Gehäuse zusammengefaßt sind, ist der CIO-Baustein AmZ8536 ein besserer Ersatz für den MC68230, wie das aus der vergleichenden Darstellung in *Tabelle 3* hervorgeht. Ein Interface, das dem des AmZ8530 (Bild 1 und 2) ähnlich ist, kann für diesen Baustein Verwendung finden.

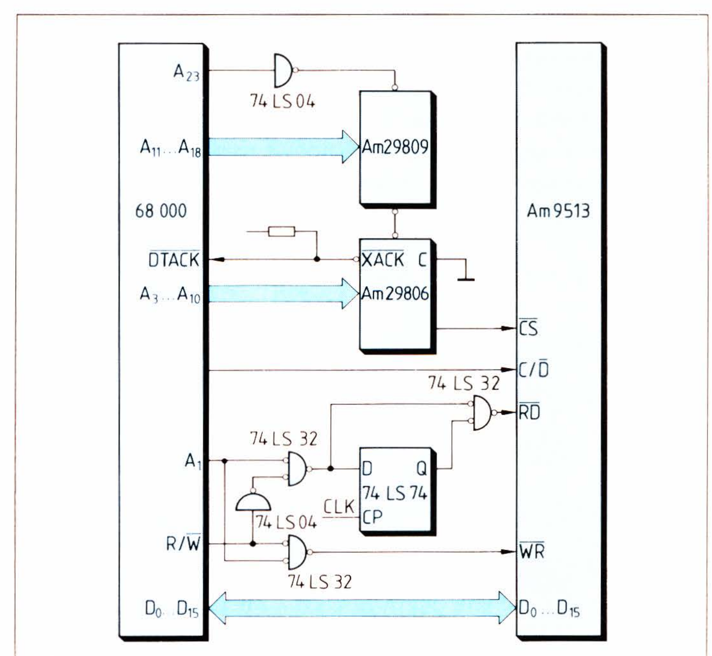


Bild 4. Der programmierbare Intervall-Zeitgeber Am9513 umfaßt fünf Zählerstufen in einem Gehäuse mit nur 40 Anschlüssen. Ein Zähler kann z. B. für die Angabe der Uhrzeit benutzt werden

4 DMA-Steuerung

4.1 *Universeller DMA-Controller Am9516*

Der universelle DMA-Controller Am9516 (UDC) ist der derzeit einzige erhältliche Hochleistungs-DMA-Baustein für Systeme, die auf der CPU 68000 basieren. Motorola hat noch keinen DMA-Chip auf den Markt gebracht, und alle verfügbaren DMA-Bausteine der ersten und zweiten Generation passen nicht zum CPU-Interface. Datentransfers werden vom Am9516 ausgeführt, indem dieser in Master-Betriebsart die Steuerung des System-Busses übernimmt. Wenn der Bus unter CPU-Steuerung läuft, ist der UDC Am9516 in Slave-Betriebsart.

Tabelle 3. Vergleich der Parallel-E/A-Bausteine

Merkmal	Z8536 CIO	MC68230 PIT	8255 PPI	8253 PIT
Betriebsspannung	+5 V	+5 V	+5 V	+5 V
Gehäuse	40pol. DIP	40pol. DIP	40pol. DIP	24pol. DIP
Technologie	N-Kanal-MOS	N-Kanal-MOS	N-Kanal-MOS	N-Kanal-MOS
Taktsignale	ein	ein	kein	drei Zähler
Anzahl der Ports	2 x 8 Bit 1 x 4 Bit	3 x 8 Bit	3 x 8 Bit	—
Anzahl der Zähler	3 x 16 Bit	1 x 24 Bit	—	3 x 16 Bit
Anzahl interner Register	Kommando, Status Daten	Datenrichtung, Status, Daten	kein	Steuerwort
Ausgangs-Tastverhältnisse	Impuls, Einzelimpuls, Rechteck	Rechteck	—	Einzelimpuls Rechteck
Programmierbare Zähler/Zeitgeber	nicht retriggerbar	vier Betriebsarten	— —	sechs Betriebsarten
Interrupt-Einrichtungen	8-Bit-Vector-Interrupt	8-Bit-Vector-Interrupt	programmierbar	programmierbar
Handshake-Operationen	interlocked, Strobe, 3-Draht und gepulst	interlocked, gepulst	Strobe	nein
Request/Wait-Signale	separate Signale für jedes Handshake-Signal	nein	nein	nein
Zähler-Betriebsarten	nur abwärts	nur abwärts	—	binär/BCD
DMA-Interface	ja	ja	nein	nein
Doppelter Puffer	ja	ja	nein	nein
Musterüberprüfung	ja	nein	nein	nein

In der DMA-Steuerschaltung (*Bild 5*) fächert der 8fach-Latch-Baustein 25LS373 den Adressen-Daten-Bus auf und speichert die Adressen. Als Zwischenspeicher für die Daten dient der Transceiver 2947. In ähnlicher Weise wird der Adreßbus $A_1...A_{23}$ vom Latch 25LS373 zwischengespeichert, obwohl lediglich ein Puffer 25LS244 erforderlich ist. Die Signale T (Transmit) und R (Receive) für den 2947 werden mit Hilfe einer PAL-Schaltung erzeugt und von den Treibern 74LS07 gepuffert.

Mit Hilfe eines Decoders 25LS138 erhält der Am9516 das CS-Signal. Ein Systemtakt synchronisiert den Betrieb der CPU, der Peripherie und des PAL-Bausteins.

Mit Hilfe des PAL-Bausteins PAL16L8 werden die Signale UDS sowie LDS aus der CPU in die DS- und B/W-Signale für den Am9516 umgesetzt, wenn die CPU 68000 sich in der Master-Betriebsart befindet. Wenn der DMA-Controller Master ist, setzt der PAL-Baustein DS, A_0 sowie B/W in die Signale UDW und LDS um. Zusätzlich wird das Ausgangssignal ALE des Am9516 in AS umbesetzt, das zum Systembus geht.

INTA wird lediglich dann erzeugt, wenn FC_0, FC_1 sowie FC_2 auf High-Potential liegen. Ein NAND-Gatter prüft diesen Zustand. Zwei D-Flipflops setzen die Signale BREQ in BR und BGACK für die CPU 68000 um. BG wird zum BACK-Signal des UDC Am9516. Den zeitlichen Verlauf der Signale zeigt *Bild 6*.

4.2 Am9516 im Vergleich mit anderen Bausteinen

Wenn DMA-Transfergeschwindigkeit und Programmierung für die speziellen Anwendungen erforderlich sind, bietet der UDC Am9516 gegenüber anderen Bausteinen, z. B. dem Typ MC6844, gewisse Vorteile:

- im Normalbetrieb bietet der Controller eine Durchsatzrate von 2,7 MByte/s (MC6844: 2,0 MByte/s). In der in „Fly-By"-Betriebsart lassen sich mit dem UDC sogar Datenraten von bis zu 4 MByte/s erreichen
- er verfügt über einen großen Umfang flexibler Betriebsarten, z. B. automatisches Aufketten, Einfügung von Wartezuständen, Software-DMA-Anforderungen, Bit-Maskierung usw.
- mit Hilfe serieller Prioritäts-Arbitrierung kann man Konzepte mit dem Am9516 auch erweitern.

DMA-Bausteine der dritten Generation werden derzeit gerade erst eingeführt (*Tabelle 4*). Der Typ MC68450 läßt einige wichtige Merkmale, die programmierbar sind, vermissen, z. B. Einfügung von Wartezuständen, Lade/Entlade-Vorgänge sowie EOP-Signale.

5 Interrupt-Controller

5.1 PIC-Baustein Am9519A

Der programmierbare Interrupt-Controller Am9519A (PIC) ist in Hochleistungssystemen anwendbar und ver-

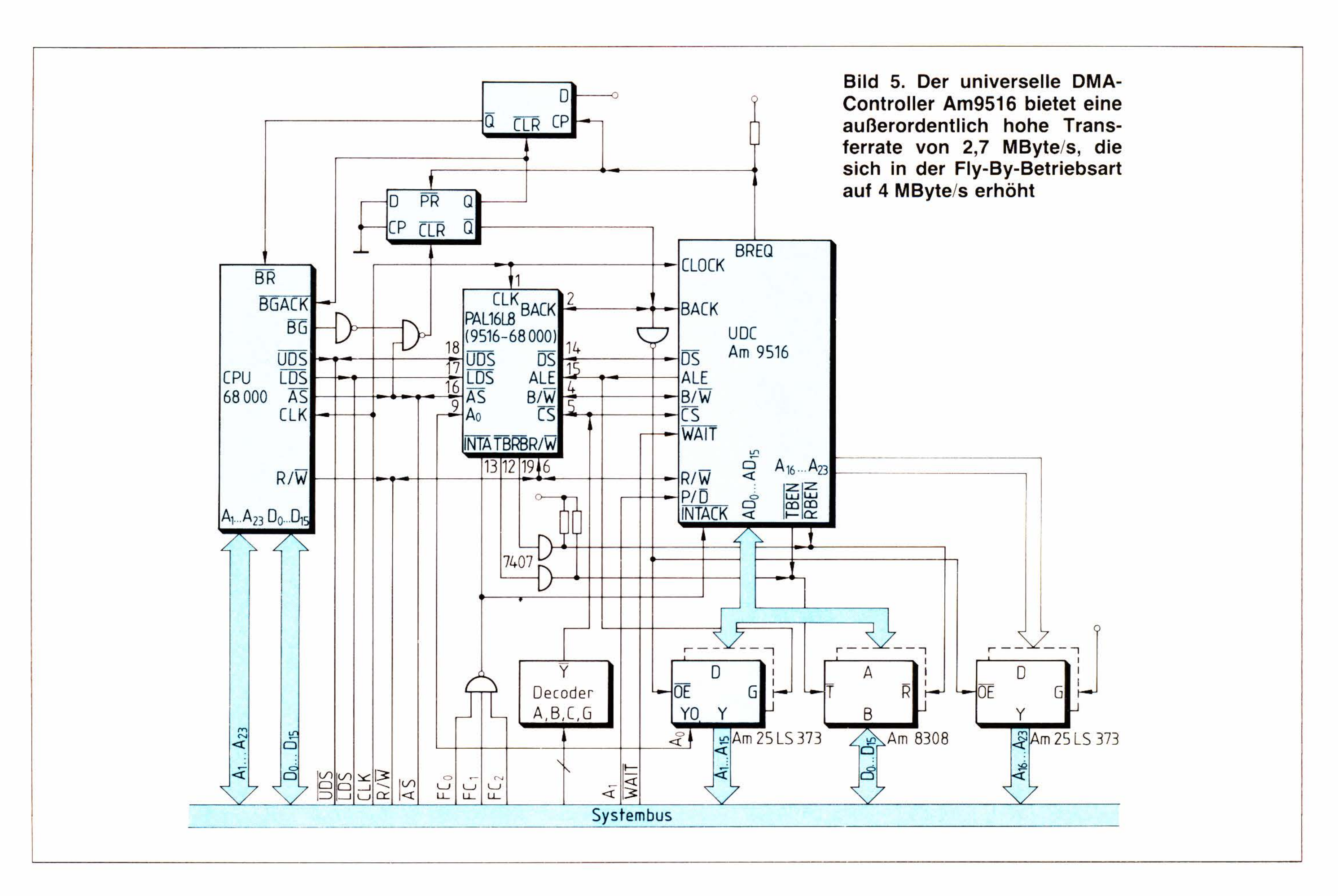


Bild 5. Der universelle DMA-Controller Am9516 bietet eine außerordentlich hohe Transferrate von 2,7 MByte/s, die sich in der Fly-By-Betriebsart auf 4 MByte/s erhöht

fügt über eine große Zahl programmierbarer Merkmale. Mit Hilfe dieses Bausteins können mehrere Interrupts priorisiert werden, wenn verschiedene Peripherieeinheiten an die CPU angeschlossen sind.

Bild 7 zeigt eine Konfiguration mit dem PIC-Baustein. Im 25LS2548 werden die Adressen decodiert und das CS-Signal für den Am9519A erzeugt. Die Datenbusleitungen $D_0...D_7$ sind direkt mit der Peripherieeinheit verbunden. Diese dient als Kommunikationspfad für die CPU, um den PIC zu programmieren sowie während der Interrupts den Vektor zurückzugeben.

Mit Hilfe des 74LS138 werden die Statussignale FC_0, FC_1 sowie FC_2 und die Interrupt-Ebenen A_1, A_2 sowie A_3 decodiert, um IACK zu generieren. Die Gatterschaltung 74LS32 arbeitet als NAND mit negativer Logik und setzt R/W sowie LDS zu den entsprechenden RD- und WR-Signalen für den Am9519A um. In ähnlicher Weise werden PAUSE und REP mit einem NAND-Gatter verknüpft, um DTACK zu erhalten. PAUSE verursacht eine bestimmte Zahl von Wartezuständen.

Das Signal GINT kann direkt angeschlossen sein, um die entsprechende Interrupt-Ebene für die CPU 68000 zu erzeugen. In das Maskenregister des Am9519A sollte nicht geschrieben werden, nachdem der Baustein freigegeben ist. Wenn der Benutzer Interrupts dynamisch freigeben oder sperren möchte, muß CS über eine UND-Funktion mit GINT verknüpft werden. Dadurch verhindert man einen irregulären Zustand, der sich ergibt, wenn ein Interrupt während des Schreibvorgangs für das Maskenregister auftritt.

Nur wenn Peripherie neben den am 9519 angeschlossenen Einheiten Verwendung findet, die ihre eigenen Vektoren liefern können, sind der Decodierer 74LS138 sowie der Codierer 74LS148 notwendig. Bei alleiniger Verwendung des Am9519 sind diese Bausteine nicht notwendig.

5.2 *Am9519 im Vergleich mit anderen Bausteinen*

Für CPU-Systeme am oberen Ende der Leistungsskala stellt der universelle Interrupt-Controller Am9519A eine leistungsfähige Struktur zur Verfügung. Dieser Baustein kann mit sehr wenig zusätzlicher Hardware an die CPU 68000 angepaßt werden. Als Interrupt-Controller für 68000-Systeme wird üblicherweise der programmierbare Intervall-Timer (PIT) MC68230 herangezogen. Dieser Baustein faßt drei Funktionen auf einem Chip zusammen: Interrupt-Controller, Intervall-Zeitgeber sowie E/A-Port. Im Gegensatz dazu ist der Typ Am9519A ein speziell für die Interrupt-Behandlung zugeschnittener Controller und bietet daher gegenüber dem MC68230 verschiedene Vorteile:

- Bausteine vom Typ Am9519A können ohne zusätzliche Hardwareunterstützung kaskadiert werden
- Vergabe der Priorität im Rotationsverfahren verhindert, daß eine Funktionseinheit die CPU blockiert
- programmierbare Vektorlänge von 1...4 Byte sorgt für jede beliebige Vektor-Adresse.

Weitere Merkmale des UIC-Bausteins sind Rücksetzmöglichkeiten während des Interrupts, unterschiedliche Bit-Maskierungs-Alternativen, zwei zusätzliche lesbare Register, die dafür sorgen, daß dieser Baustein sich an die unterschiedlichsten Systemumgebungen anpassen läßt. Falls drei Funktionen auf einem Chip zusammenzufassen sind, ist der Typ AmZ8586 ein besserer Ersatztyp für den Baustein MC68230. Dazu ist ein Interface wie beim AmZ8530 (Bild 1 und 2) erforderlich.

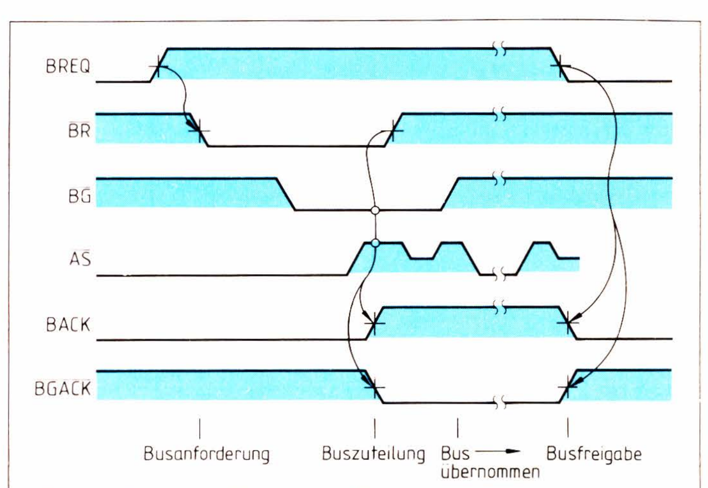


Bild 6. Zeitlicher Ablauf der Buszuteilung für den Master-Slave-Betrieb des DMA-Controllers

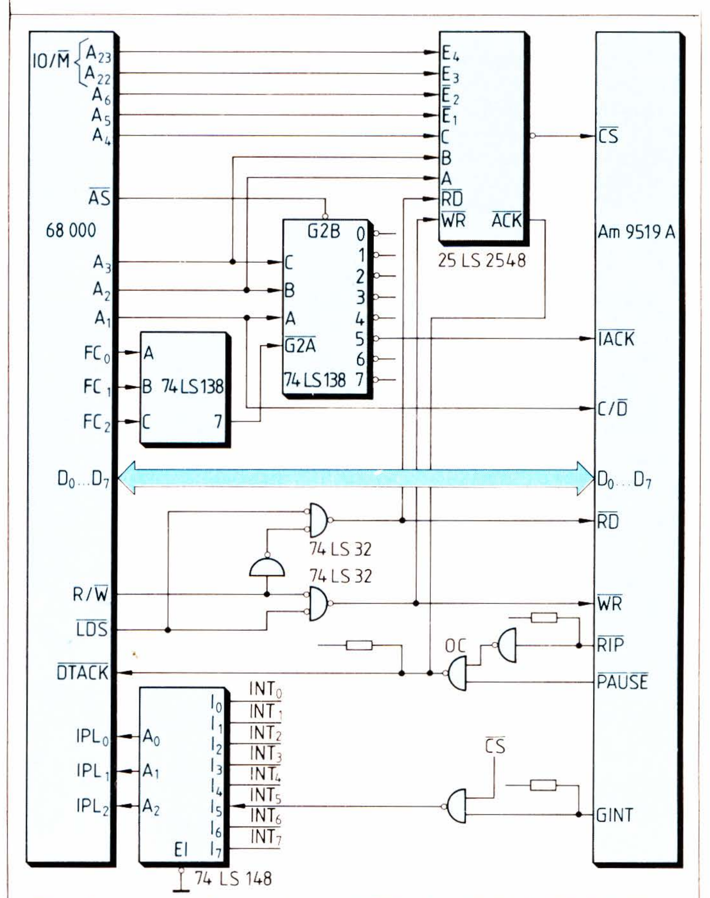


Bild 7. Interrupt-Controller Am9519A in Verbindung mit der CPU 68000. Dieser Baustein läßt sich ohne zusätzliche Hardware zu erweiterten Systemen kaskadieren

Tabelle 4. Übersicht der DMA-Controller

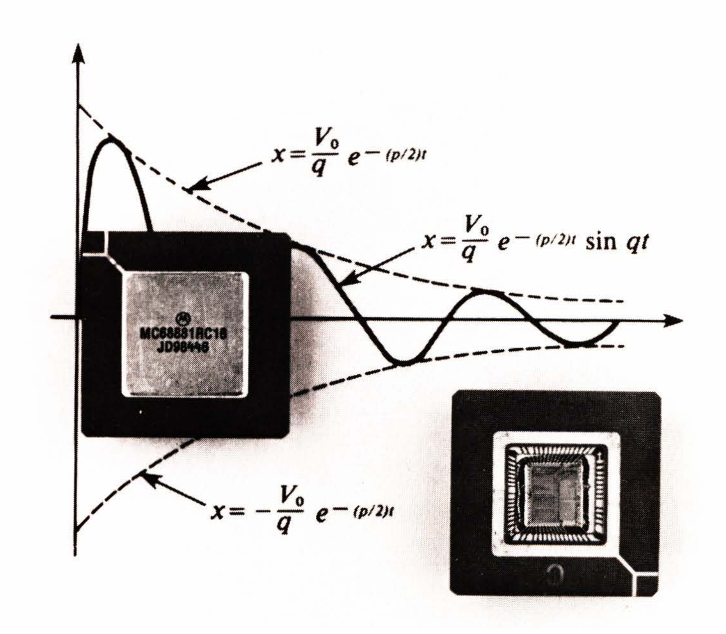
Merkmal	AMD9517A-5	Am9516	AMD8016	Intel 8089	MC68450
Betriebsspannung	+5 V	+5 V	+5 V	+5 V	+5 V
Gehäuse	40pol. DIP	48pol. DIP	48pol. DIP	40pol. DIP	64pol. DIP
Technologie	N-Kanal-MOS	N-Kanal-MOS	N-Kanal-MOS	N-Kanal-MOS	N-Kanal-MOS
Taktsignale	ein	ein	ein	ein	ein
Anzahl der Kanäle	vier	zwei	zwei	zwei	vier
Adreßbereich	physikalisch 64 K	physikalisch 16 M	8 M logisch, 16 M physikal.	physikalisch 1 M	physikalisch 16 M
Blockgröße	64 K	16 M	8 M...16 M	1 M	?
Minimale Taktzykluszeit	200 ns	167 ns	250 ns	200 ns	?
maximale Transferrate	1,66 MByte/s (9517-4: 2 M)	2,66 MByte/s	2,66 MByte/s	1,25 MByte/s	4 MByte/s
Automatisches Lader der Basisadresse	separate Basisregister für jeden Kanal	separate Basisregister für jeden Kanal	separate Basisregister für jeden Kanal	nein	separate Basisregister für jeden Kanal
feste oder rotierende Priorität	ja	nein	nein	ja	ja
Maskierbare Anforderung	ja	ja	ja	ja	nein
direkte Steuerung individueller Maskenbits	ja	ja	ja	nein	nein
Betriebsart-Register	separates 6-Bit-Mode-Register für jeden Kanal	8-Bit-Master-Mode-Register	8-Bit-Master-Mode-Register	20-Bit-Register pro Kanal	separat
separate DMAREQ- u. DMAACK-Leitungen	ja	ja	ja	nein	ja
Software-DMA-Request	ja	ja	ja	ja	?
Status-Register auf dem Chip	ja	ja	ja	ja	ja
Speicher-Speicher-Transfer	ja	ja	ja	ja	ja
Increment/Decrement von Kanaladressen	ja	ja	ja	ja	ja
Blockweises Setzen von Speicherbereichen auf Null oder Konstante	ja	ja	ja	ja	ja
Arbeitsweise der Acknowledge- und Request-Leitungen	programmierbar	programmierbar	programmierbar	fest	?
Transfer-Betriebsarten	Einfach, Demand Block	Einfach, Demand	Einfach, Demand	Block	Einfach, kontinuierlich kettenförmig
erweitertes Schreiben	ja	—	—	nein	nein
Ende-Signal zum Abbruch des Transfer	ja	ja	ja	ja	nein
DMA-Operationen	Read, Write, Verify	Transfer, Suchen, Transfer und Suche	Transfer, Suchen, Transfer und Suche	Lesen, Schreiben Suchen, Vergleichen, Transfer	Lesen, Schreiben
Erweiterung	Kaskadierung	serielle Prioritäts-Arbitration	serielle Prioritäts-Arbitration	nein	nein
Prioritäts-Arbitration zwischen DMA-Controllern	übergibt Haltaufforderung nach jedem Transfer	ja	ja	ja	nein
programmierbare Wartezustände	nein	automatisches Einfügen von 0, 1, 2 oder 4 Wartezuständen	automatisches Einfügen von 0, 1, 2 oder 4 Wartezuständen	ja	nein
Interrupt-Einrichtungen	nein	Vektor-Interrupts bei bestimmten Transfers	Vektor-Interrupts bei bestimmten Transfers	nein	programmierbar
Steuerparameter laden/neu laden	CPU-Intervention erforderlich	automatisches Verketten aufeinanderfolgender Transfers	automatisches Verketten aufeinanderfolgender Transfers	CPU-Intervention erforderlich	erfordert externes RAM

Dipl.-Ing. (FH) Werner Hilf

Gleitkomma-Arithmetik-Coprozessor

Ein schneller Rechenkompagnon für Mikroprozessoren

Die Aufgaben, die Mikroprozessoren zu erledigen haben, sind längst über „Ja/Nein-Entscheidungen" und einfache Ablaufsteuerungen hinausgewachsen. Sie müssen mittlerweile wahre „Rechengenies" sein. Bisher wurden diese Rechenaufgaben softwaremäßig gelöst, was jedoch zu längeren Rechenzeiten führte. Deshalb hat man die Rechenprogramme „in Silizium gegossen", d. h. spezielle Arithmetikprozessoren entwickelt. Diese Bausteine arbeiten, wenn sie einmal „gestartet" wurden, parallel zum Hauptprozessor und sind deren „Kompagnons" (sog. Coprozessoren).



Um alle Funktionen des Gleitkomma-Coprozessors auf einem Chip unterzubringen, mußte man sich einer neuen Technologie bedienen: Der HCMOS-Technologie, die bereits beim 32-Bit-Mikroprozessor erprobt wurde. Sie ermöglicht, daß der Baustein (zunächst) mit einer Taktfrequenz von 16,67 MHz betrieben werden kann; die Verlustleistung beträgt rund 1,5 W

Der Gleitkomma-Arithmetikprozessor MC 68881 von Motorola eignet sich zum Anschluß an die Mikroprozessoren der Familie 68000 [1]. Ab Mitte dieses Jahres wird er lieferbar sein. Daß man so lange auf die Gleitkomma-Prozessoren warten mußte, lag einerseits an der recht „zähen" Standardisierung der Datenformate durch IEEE (P754) und andererseits an den hohen Anforderungen an die Halbleiter-Technologie. Die Standardisierung wurde bereits 1981 begonnen, machte aber, wie üblich, einige Revisionen durch und liegt nun „endgültig" vor (Revision 10.0). Der Coprozessor MC 68881 wurde nach diesem Standard entwickelt und enthält alle Funktionen.

Coprozessoren sollen die Zentraleinheit entlasten; sie können speziell für ihre Aufgaben „gezüchtet" werden und müssen deshalb keine Allround-Könner sein. Man unterscheidet dabei zwei Arten von Kompagnons: Erstens Coprozessoren, die für eine spezielle CPU geschaffen wurden (sie horchen sozusagen den Bus des Mikroprozessors ab und picken die für sie bestimmten Befehle heraus); zweitens Coprozessoren, die wie Peripheriebausteine angesprochen werden (sie sind über Adreß-, Daten- und Funktionsleitungen an die CPU angeschlossen). Der MC 68881 gehört zu der zweiten Kategorie und verfügt über einen eigenen Chip-Select-Anschluß (*Bild 1*).

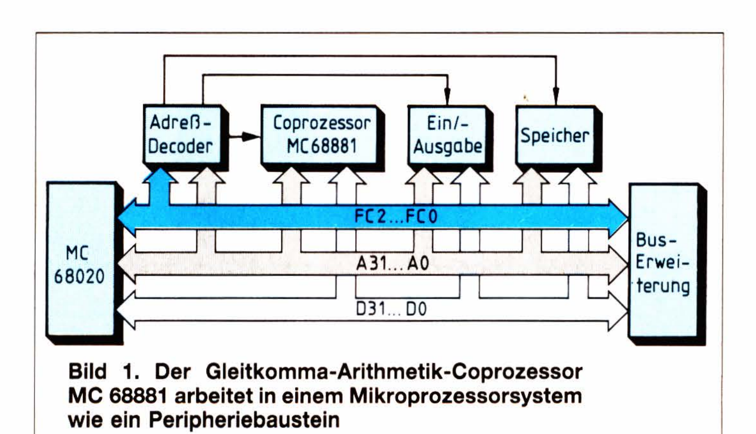


Bild 1. Der Gleitkomma-Arithmetik-Coprozessor MC 68881 arbeitet in einem Mikroprozessorsystem wie ein Peripheriebaustein

Das „Rechengenie" im System

Der 32-Bit-Mikroprozessor MC 68020 besitzt ein integriertes Coprozessor-Interface, so daß

der Arithmetik-Baustein MC 68881 einfach an diese CPU angeschlossen werden kann. Aber auch die anderen Prozessoren der Familie 68000 können mit dem „Rechengenie“ zusammenarbeiten. Ermöglicht wird dies durch die integrierte dynamische Busstruktur (*Bild 2*).

Der Anschluß des MC 68020, für den der Arithmetikprozessor entwickelt wurde, erfolgt ohne Software- und größeren Hardwareaufwand. Bei den anderen Familienmitgliedern muß die Coprozessor-Kommunikation per Software nachgebildet werden. Glücklicherweise haben die Entwickler der Mikroprozessoren einen Freiraum für zukünftige Befehle gelassen. Dieser ist bekannt als sog. Line-F-Emulator [2]. Alle Befehle, die hexadezimal mit FXXX beginnen (der Opcode umfaßt 16 Bit), sind nicht implementiert und führen zu „Exceptions“. Der Mikroprozessor MC 68020 versteht die Coprozessor-Befehle, die alle mit FXXX beginnen (*Bild 3*). Die Bits 9 bis 11 enthalten die Coprozessor-Nummer; denn in einem 68000-System können bis zu acht Coprozessoren vorhanden sein. Die Bits 6 bis 8 unterscheiden die Coprozessor-Befehlstypen („Primitives“):

- ○ allgemein (alle mathematischen Funktionen)
- ○ bedingte Sprünge (branch on condition)
- ○ bedingter „Fang“ (trap on condition)
- ○ bedingtes Setzen (set on condition)
- ○ Dekrementieren und bedingter Sprung (decrement and branch on condition)
- ○ Retten des Coprozessor-Zustandes (save)
- ○ Rückspeichern des Zustandes bei Taskumschaltung und virtueller Speicherverarbeitung (restore).

Zur Spezifikation einer eventuellen Operanden-Adressierungsart dienen die Bits 0 bis 5 [2], während in den nachfolgenden Worten der eigentliche Coprozessor-Befehl oder die Operanden folgen. Bei allen diesen

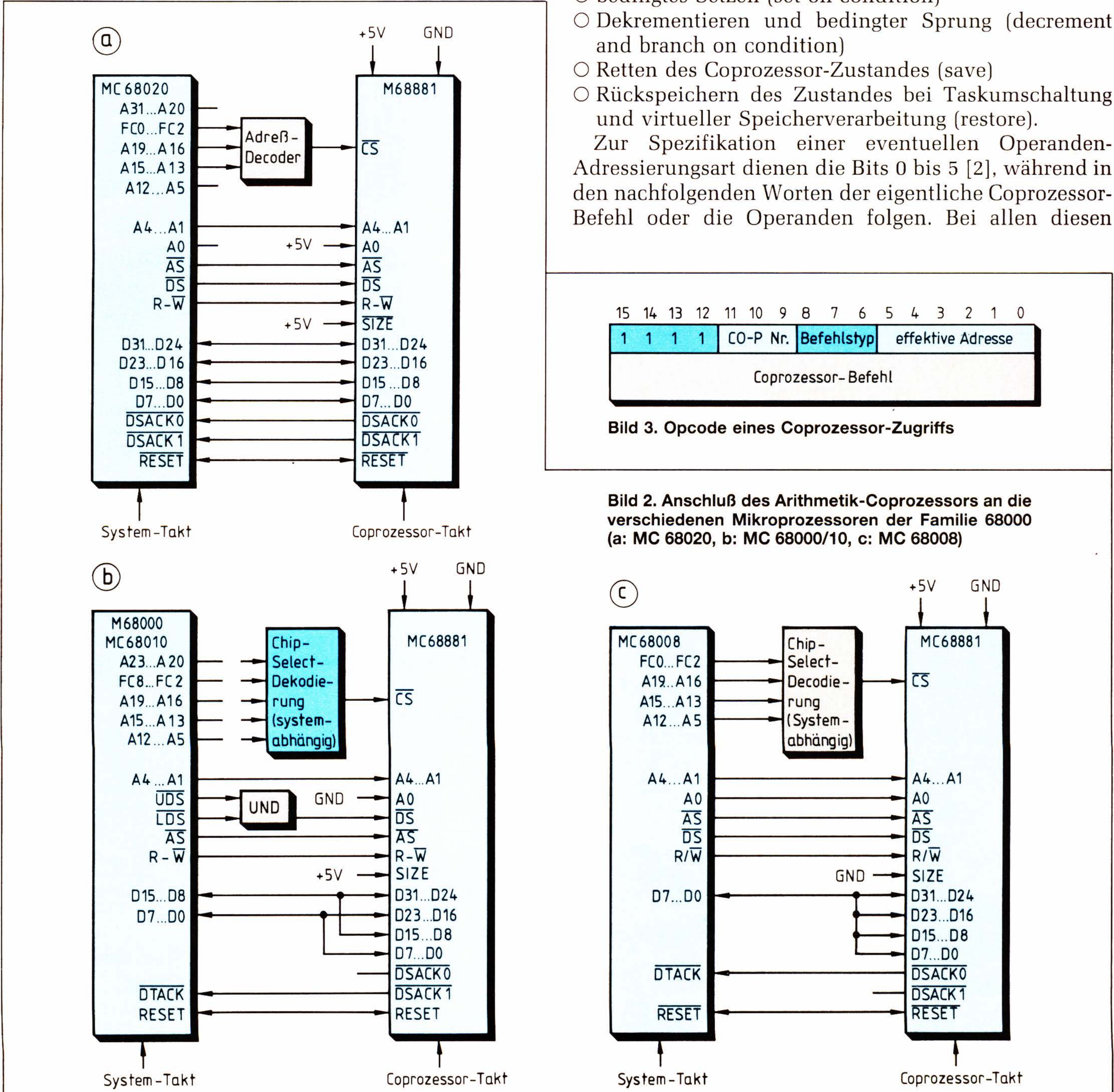


Bild 3. Opcode eines Coprozessor-Zugriffs

Bild 2. Anschluß des Arithmetik-Coprozessors an die verschiedenen Mikroprozessoren der Familie 68000 (a: MC 68020, b: MC 68000/10, c: MC 68008)

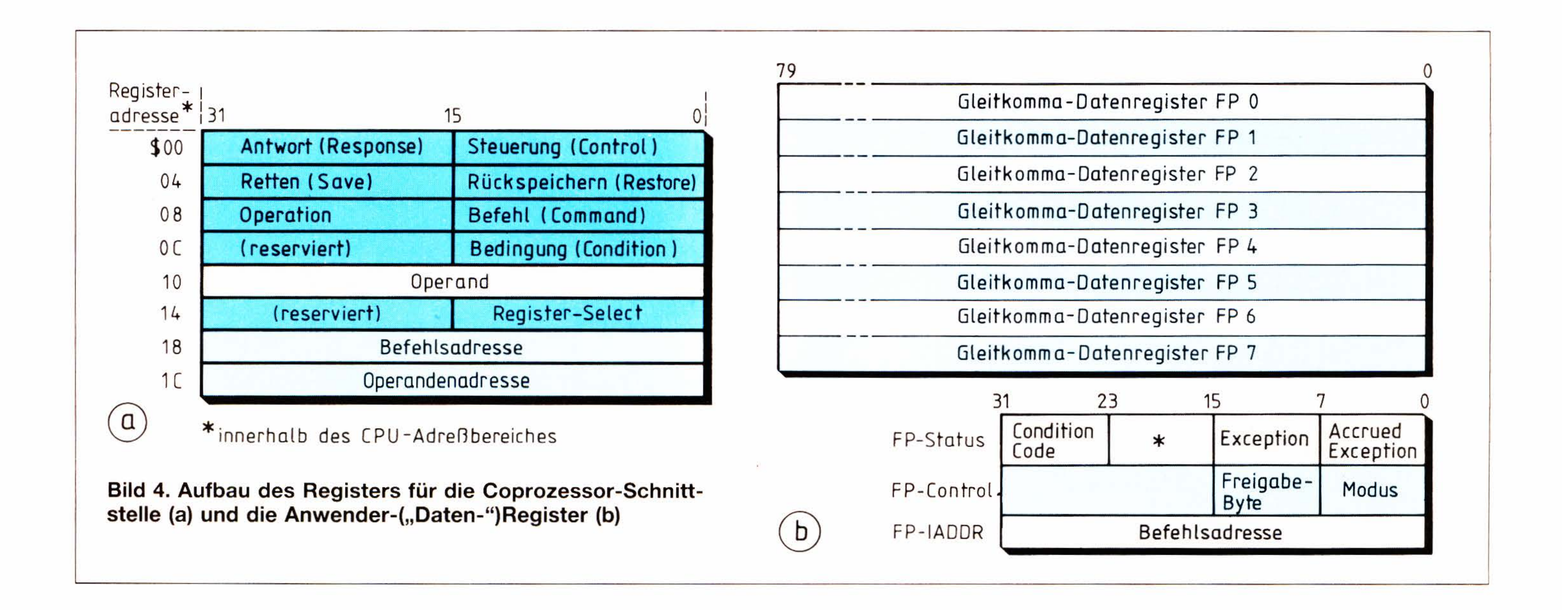


Bild 4. Aufbau des Registers für die Coprozessor-Schnittstelle (a) und die Anwender-(„Daten-")Register (b)

Befehlen des MC 68020 werden zusätzlich, wie bereits in [1] beschrieben, die Funktionscode-Leitungen FC0 bis FC2 auf „1" gesetzt. Damit liegen die Adressen der Coprozessor-Register in einem eigenen Adreßbereich („CPU-Space").

Bild 4a zeigt die Coprozessor-Schnittstellen-Register. Im Response-(Antwort-)Register werden die nächsten Aktionen definiert, die der Coprozessor erwartet, bzw. die der Hauptprozessor ausführen soll. Es ist eines der wichtigsten Bindeglieder in der Kommunikationskette zwischen Haupt- und Coprozessor. Letzterer kann u. a. folgende Zustände annehmen:

- beschäftigt
- weiter mit dem nächsten Befehl
- lese oder schreibe Daten von bzw. zur effektiven Adresse
- transferiere Programmzähler (PC) der CPU zum Coprozessor
- transferiere Statusregister (SR) der CPU zum Coprozessor
- transferiere Befehlswort
- errechne effektive Adresse und transferiere sie
- transferiere weitere Operanden
- bedingtes Setzen/bedingter Sprung erfüllt/nicht erfüllt
- nochmals abfragen (CA = „Come Again")

Die Coprozessorbefehle (im Mikrocode integriert) des MC 68020 arbeiten automatisch mit diesen Registern zusammen. Die anderen Mikroprozessoren der Familie 68000 müssen die Antwort des Arithmetik-Prozessors abfragen bzw. die Coprozessorbefehle, die der Assembler generiert hat, im Line-F-Exception-Programm decodieren. Die genaue Funktion der einzelnen Bits der verschiedenen Register kann in [4] nachgelesen werden.

Die Kommunikation zwischen Haupt- und Coprozessor endet, wenn letzterer alle zur Ausführung eines Befehls nötigen Informationen (z. B. Kommandos, Operanden) erhalten hat. Daraufhin arbeitet der Coprozessor selbständig, und der Hauptprozessor kann ungestört andere Aufgaben erledigen.

Die Anschlüsse des Gleitkomma-Prozessors haben eine gewisse Ähnlichkeit mit denen des MC 68020; es gibt jedoch nur fünf Adreßleitungen, die zur Auswahl der Coprozessor-Schnittstellen-Register dienen (*Tabelle 1*). Die Adreßleitung A0 dient zusammen mit der SIZE-Leitung zur Bestimmung der Datenbusbreite (*Tabelle 2*). Diese wird, je nach verwendetem Hauptprozessor, fest eingestellt.

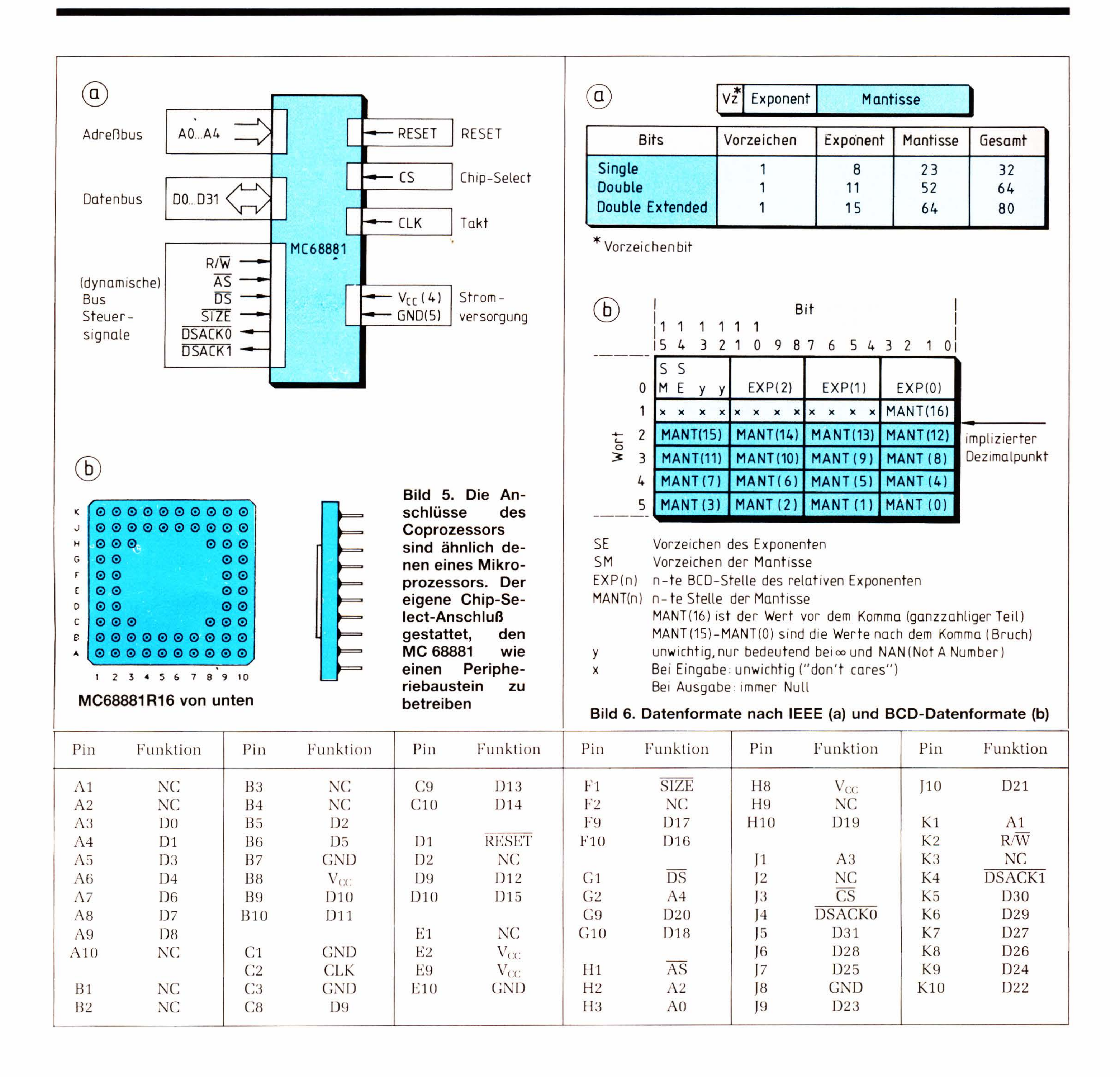
Da es sich um ein asynchrones System handelt, kann die Taktfrequenz (CLK) von der des Hauptprozessors verschieden sein – sie muß es aber nicht sein. Das Chip-Select-Signal ($\overline{CS}$) deutet an, daß der Baustein wie ein peripheres Bauelement zu behandeln ist (*Bild 5*).

Tabelle 1. Coprozessor-Interface-Registerauswahl

Adreß-leitung A4...A0	Offset Hex	Länge Bits	Zugriff	Coprozessor Interface-Register
0000x	00	16	Read	Response
0001x	02	16	Write	Control
0010x	04	16	Read	Save
0011x	06	16	R/W	Restore
0100x	08	16	–	(Reserved)
0101x	0A	16	Write	Command Word
0110x	0C	16	–	(Reserved)
0111x	0E	16	Write	Condition Word
100xx	10	32	R/W	Operand
1010x	14	16	Read	Register Selector
1011x	16	16	–	(Reserved)
110xx	18	32	R/W	Instruction Address
111xx	1C	32	–	(Reserved)

Tabelle 2. Festlegen der Datenbusbreite

$\overline{Size}$	A0	Konfiguration
1	1	32 Bit
1	0	16 Bit
0	–	8 Bit



Bits	Vorzeichen	Exponent	Mantisse	Gesamt
Single	1	8	23	32
Double	1	11	52	64
Double Extended	1	15	64	80

Bild 5. Die Anschlüsse des Coprozessors sind ähnlich denen eines Mikroprozessors. Der eigene Chip-Select-Anschluß gestattet, den MC 68881 wie einen Peripheriebaustein zu betreiben

Bild 6. Datenformate nach IEEE (a) und BCD-Datenformate (b)

Pin	Funktion	Pin	Funktion	Pin	Funktion	Pin	Funktion	Pin	Funktion	Pin	Funktion
A1	NC	B3	NC	C9	D13	F1	$\overline{SIZE}$	H8	V_{CC}	J10	D21
A2	NC	B4	NC	C10	D14	F2	NC	H9	NC		
A3	D0	B5	D2			F9	D17	H10	D19	K1	A1
A4	D1	B6	D5	D1	$\overline{RESET}$	F10	D16			K2	R/$\overline{W}$
A5	D3	B7	GND	D2	NC			J1	A3	K3	NC
A6	D4	B8	V_{CC}	D9	D12	G1	$\overline{DS}$	J2	NC	K4	$\overline{DSACK1}$
A7	D6	B9	D10	D10	D15	G2	A4	J3	$\overline{CS}$	K5	D30
A8	D7	B10	D11			G9	D20	J4	$\overline{DSACK0}$	K6	D29
A9	D8			E1	NC	G10	D18	J5	D31	K7	D27
A10	NC	C1	GND	E2	V_{CC}			J6	D28	K8	D26
		C2	CLK	E9	V_{CC}	H1	$\overline{AS}$	J7	D25	K9	D24
B1	NC	C3	GND	E10	GND	H2	A2	J8	GND	K10	D22
B2	NC	C8	D9			H3	A0	J9	D23		

Befehlssatz des „Rechengenies"

Der Gleitkomma-Arithmetikprozessor versteht nicht nur mathematische Befehle, sondern kann auch bedingte Sprünge durchführen sowie Werte in Abhängigkeit von Bedingungen setzen.

Ein Beispiel:

68000-Befehl (ganzzahlig)	68881-Befehl (Gleitkomma)
CMP D0,D1	FCMP FP0, FP1
BEQ LOOP	FBEQ LOOP

Reichen die acht 80-Bit-„Arbeitsregister" FP0 bis FP7 des Coprozessors (*Bild 4b*) nicht aus, so können diese abgespeichert und zurückgeholt werden (FMOVEM). Bei einer Taskumschaltung (timeslice) muß man z. B. den Zustand des MC 68881 ähnlich wie beim BERR des Mikroprozessors retten bzw. zurückholen. Hierzu werden die Befehle SAVE oder RESTORE benutzt. Oft benötigt man bei Berechnungen Konstanten (π, e usw.), die mit Hilfe des Befehles FMOVECR aus einem internen ROM geholt werden.

Im Statusregister stehen die Flags, die bei Gleitkomma-Operationen manipuliert werden, und auf die bei den Gleitkommaabfragen Bezug genommen wird (z. B. FB_{CC}). Im Controlregister sind die genormten Rundungsarten (IEEE P754) festgelegt.

Tabelle 3. Zusammenfassung der Befehle

Befehl	Bedeutung
1) *MOVE-Befehle*	
FMOVE	MOVE von und zum FP-Datenregister (FPn), Control-Register (CR), Status-Register (SR) und
FMOVECR	MOVE von Konstanten ROM (z. B. Zahl π)
FMOVEM	MOVEM von und zu FPn, SR, CR und IADDR
2) *bedingte Befehle*	
FBcc	bedingter Sprung
FDBcc	bedingter Sprung mit Dekrement
FScc	bedingtes Setzen
FTcc	bedingter TRAP
FTPcc	bedingter TRAP mit Parameter
3) *ein Argument*	
(monadic)	
FABS	absoluter Wert
FGETEXP	Exponent holen
FGETMAN	Mantisse holen
FINT	Ganzzahligen Teil holen
FNEG	Negation
FSCALE	Skalierung 2^n
FSQRT	Wurzel
FTEST	Test
4) *zwei Argumente*	
(dyadic)	
FADD	Addition
FCMP	Vergleich
FDIV	Division
FMOD	Modulo-Rest
FMUL	Multiplikation
FREM	IEEE-Rest
FSUB	Subtraktion
5) *transzendente Befehle*	
Befehle	
FACOS	arccos
FASIN	arcsin
FATAN	arctan
FATANH	arctanh
FCOS	cos
FCOSH	cosh
FETOX	e^x
FETOXM1	e^x-1
FLOG10	$\log_{10}$
FLOG2	$\log_2$
FLOGN	$\log_n$
FLOGNP1	$\log_n (x+1)$
FSIN	sin
FSINCOS	sin und cos gleichzeitig berechnen
FSINH	sinh
FTAN	tan
FTANH	tanh
FTENTOX	10^x
FTWOTOX	2^x
6) *Verschiedenes*	
FSGLMUL	Single Precision (32 Bit) Multiplikation
FSGLDIV	Single Precision (32 Bit) Division
FSAVE	Retten des Coprozessorzustandes
FRESTORE	Rückspeichern des Coprozessorzustandes

Tabelle 4. Ausführungszeiten der wichtigsten Befehle

	Register/Register		Speicher/Register					
			Single		Double		Extended	
Befehl	Takt-zyklen	Zeit [µs]	Takt-zyklen	Zeit [µs]	Takt-zyklen	Zeit [µs]	Takt-zyklen	Zeit [µs]
FMOVE In	26	1.5	51	3.0	57	3.4	55	3.3
FADD/FSUB	46	2.8	71	4.3	77	4.6	75	4.5
FSGLMUL	52	3.1	77	4.6	–		–	
FSGLDIV	64	3.8	89	5.3	–		–	
FMUL	66	4.0	91	5.5	97	5.8	95	5.7
FDIV	98	5.9	123	7.4	129	7.7	127	7.6
FSIN	384	23.0	409	24.5	415	24.9	413	24.8
FCOS	384	23.0	409	24.5	415	24.9	413	24.8
FSINCOS	426	25.5	451	27.0	457	27.4	455	27.3
FTAN	454	27.2	479	28.7	485	29.1	483	29.0
FASIN	564	33.8	589	35.3	595	35.7	593	35.6
FACOS	598	35.9	623	37.4	629	37.7	627	37.6
FATAN	378	22.7	403	24.2	409	25.5	407	24.4

Der komplette Befehlssatz ist in *Tabelle 3* zusammengefaßt. Bei allen Befehlen, außer MOVE, SAVE und RESTORE, muß das „Ziel“ grundsätzlich eines der acht Gleitkomma-Register sein. Die „Quelle“ kann eine effektive Adresse oder ein Gleitkomma-Register sein.

Geschwindigkeit ist keine Hexerei

Eines der wichtigsten Kriterien für den Einsatz eines Gleitkomma-Prozessors ist die Rechengeschwindigkeit. In *Tabelle 4* stehen die einzelnen Ausführungszeiten, sie enthalten auch alle mit dem Coprozessor zusammenhängenden Overhead-Zeiten. Umwandlungsroutinen sind nicht erforderlich.

Mit der Kombination aus Mikroprozessor MC 68020 und Arithmetik-Coprozessor MC 68881 können folgende Datentypen verarbeitet werden (*Bild 6*):

- ○ Byte (8 Bit) Integer (B)
- ○ Wort (16 Bit) Integer (W)
- ○ Langwort (32 Bit) Integer (L)
- ○ Single Precision Binär (32 Bit) Real (S)
- ○ Double Precision Binär (64 Bit) Real (D)
- ○ Extended Precision Binär (80 Bit) Real (X)
- ○ Packed Dezimal (96 Bit) Real (P).

Diese Datentypen sind in allen Befehlen anwendbar. Beispiele sind:

FINT.S ARG1,FP3	Ganzzahligen Teil von ARG1 bilden
FABS.X FP2,FP2	Absoluten Betrag des Inhaltes des Gleitkomma-Registers FP2 bilden (80 Bit Real)
SIN.D 3(A0), FP2	Sinus bilden von effektiver Adresse 3(A0), Ergebnis nach FP2
FATAN.B #5, FP6	Arctan von 5 bilden, Ergebnis steht in FP6
FADD.P #4.012E+15,FP3	Addition einer Real-Dezimalzahl zu FP3

```
                 Y:    = IX  +  YPS(I)  +  ZET  *  5
Variable         ↑       ↑        ↑         ↑      ↑
vom Typ:         P       S        D         L      B

FMOVE.S   (A7), FP0                Hole IX aus Stack
FADD.D    ([10,A7], D0.W*8),FP0    Addiere IX+YPS(I)
FADD.L    D3,FP0                   Addiere ZET
FMUL.B    #5,FP0                   Multipliziere mit 5
FMOVE.P   FP0,-(A7)                Lege das Ergebnis auf den
                                   Stack ab (und wandle um)
```

Bild 7. Umsetzung eines Befehls in MC 68020/68881-Mnemonik

Alle Operanden werden in 80-Bit-Werte umgewandelt. Um die automatischen Umwandlungen zu erklären, sei folgendes Beispiel gegeben: Drei Variable (IX, YPS, ZET) unterscchiedlicher Datentypen sollen addiert und anschließend mit der Konstanten „5“ multipliziert werden. Das Ergebnis (Y) soll vom Typ „Packed Decimal Real“ sein. Die Variable IX liegt im Stack und die Variable YPS in einer Tabelle, die indirekt über einen Zeiger (Pointer), der ebenfalls im Stack abgespeichert ist, angesprochen wird. Die Variable ZET befindet sich im Datenregister D3 des 32-Bit-Prozessors, und das Ergebnis soll wiederum im Stack abgelegt werden (*Bild 7*). Die Umwandlung der Variable IX erfolgt bereits während des Holens aus dem Stack bzw. während der Ablage des Ergebnisses in den Stack.

Hier ist es also gelungen, den in Wahrheit getrennten Coprozessor als Teil des Hauptprozessors erschienen zu lassen (Transparenz). Die Anwenderregister des Coprozessors stellen so eine logische Erweiterung des Registermodells im Hauptprozessor dar.

Literatur

[1] *Hilf, W.*: MC 68020 – 32-Bit-Prozessor für zukunftssichere Systeme. ELEKTRONIK 1984, H. 14, S. 35...42.

[2] *Hilf, W.; Nausch, A.*: M68000, Teil 1 (Grundlagen und Architektur). tewi-Verlag, München.

[3] *Hilf, W.; Nausch, A.*: M68000, Teil 2 (Anwendung und 68000-Bausteine). tewi-Verlag, München.

[4] Motorola, Inc.: MC68020 32-Bit Virtual Memory Microprocessor – Users Manual. 1. Ausgabe, Mai 1984. Prentice-Hall, ISBN 0-13-541467-9.

[5] *Huntsman, C.; Cawthron, D.*: The MC 68881 Floating Point Coprocessor. IEEE MICRO, Dez. 1983.

Werner Hilf ist in Schopfheim/Krs. Lörrach geboren. Er hat die Elektronik „von der Pike auf“ gelernt. 1967 Lehre in einem Entwicklungs- und Forschungslabor. Studium der Nachrichtentechnik 1970 an der Fachhochschule Karlsruhe. Zunächst Analog-Elektronik, insbesondere Prozeß- und Regeltechnik. Mit dem Siegeszug des Mikroprozessors und fasziniert von der Digitaltechnik begann er mit vielfältigsten Aufgaben aus dem Bereich der Hard- und Software mit diversen Mikroprozessoren. Seit 1979 ist er Leiter der Mikroprozessorschulung bei Motorola in München. Zahlreiche Vorträge im Inland, europäischen Ausland und Afrika. Sein Wissen vermittelt er seit 1980 auch als Lehrbeauftragter an der Fachhochschule München und der Technischen Akademie in Esslingen.

Prof. Dipl.-Ing. Günter Schmitt

68000-Computer zum Kennenlernen

Der Einstieg in die Mikrocomputertechnik geschieht in der Regel mit Hilfe eines Übungscomputers, der neben dem Prozessor einen Festwertspeicher mit dem Monitorprogramm, einen kleinen Arbeitsspeicher und eine Schnittstelle für die Ein-/Ausgabe enthält. Der Monitor ist ein Programm, das die Ein-/Ausgabe-Operationen ermöglicht und Hilfen für den Test der Benutzerprogramme bereitstellt. Diese Art der Einführung in die Mikrocomputertechnik ist für den Anfänger leicht verständlich und zeigt die Verbindung zur Hardware; sie hat sich im Unterricht und im Selbststudium bewährt. Die vorliegende Arbeit beschreibt einen einfachen Mikrocomputer mit dem 16-Bit-Prozessor 68000, der sich auf einer Europakarte einfach aufbauen läßt. Er kann in Fädeltechnik mit Kupferlackdraht verdrahtet werden. Um dem Leser den Einstieg in die Programmierung des Prozessors zu erleichtern, folgen einige weitere Beiträge mit Monitorprogrammen für Terminal- und Tastaturbetrieb sowie Befehlslisten und Programmbeispiele.

1 Aufgabenstellung

Es sollte ein Mikrocomputer mit dem 16-Bit-Mikroprozessor 68000 entwickelt werden, der sich auf einer Europakarte aufbauen läßt und der aus preiswerten und handelsüblichen Bausteinen besteht. *Bild 1* zeigt die Übersichtsschaltung.

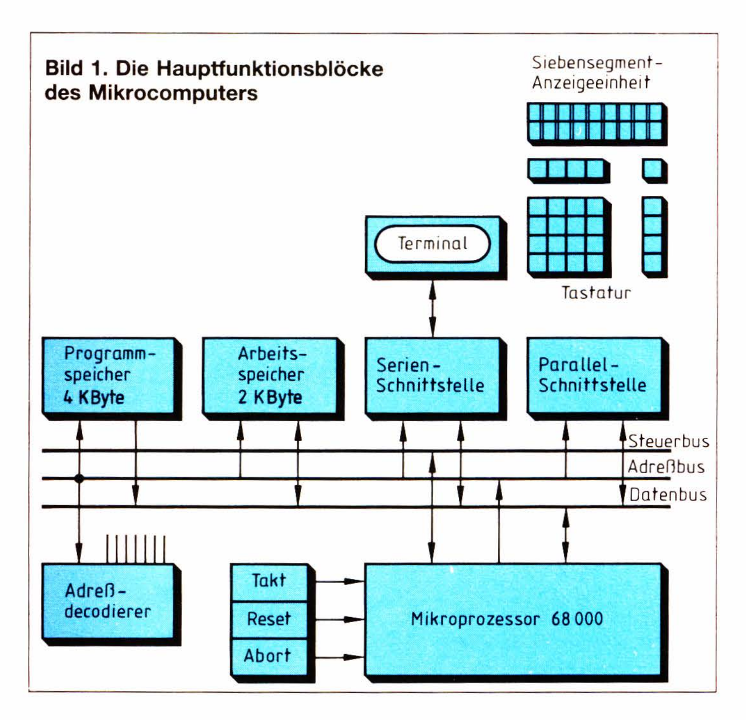


Bild 1. Die Hauptfunktionsblöcke des Mikrocomputers

Ein Taktgenerator versorgt den Prozessor mit einem Takt von 4 MHz für die langsamste Ausführung des Prozessors. Mit Reset wird der Monitor gestartet; die Abort-Taste löst einen Interrupt aus und bricht damit ein Benutzerprogramm ab.

Zwei EPROM-Bausteine vom Typ 2716 bilden einen 4-KByte-Programmspeicher für ein kleines Monitorprogramm. Wegen des 16 Bit breiten Datenbusses müssen zwei Bausteine parallel betrieben werden. Vier RAM-Bausteine vom Typ 2114 bilden einen 2-KByte-Arbeitsspeicher, von dem der Monitor etwa 128 Bytes belegt; der Rest steht dem Benutzer zur Verfügung.

Für den Betrieb mit dem Tastenmonitor stellt eine Parallelschnittstelle 6821 die Verbindung zur Hexadezimaltastatur und der Siebensegment-Anzeige her. Der Betrieb mit dem Terminalmonitor erfolgt über eine Serien-Schnittstelle 6850 mit einem Taktgenerator und V.24-Treibern.

2 Der Prozessor

Dieser Abschnitt beschreibt im wesentlichen nur die wichtigsten Eigenschaften des Prozessors, die in der ausgeführten Schaltung verwendet wurden. Das Benützer-Handbuch [1] und die Datenblätter [2] des Herstellers enthalten die vollständige Beschreibung. Eine weitere Einführung ist in [4]...[6] enthalten. *Bild 2* zeigt den Registersatz und die Anschlüsse nach Funktionen geordnet.

2.1 Der Registersatz

Der Prozessor arbeitet intern mit 32 Bit breiten Registern; fast alle Befehle lassen sich auf Operanden von 32 Bit anwenden. Die acht Datenregister D0...D7 dienen als Akkumulatoren; die sieben Adreßregister A0...A6 werden für die Behandlung von Adressen als Indexregister und Stapelzeiger verwendet. Das Adreßregister A7 ist doppelt vorhanden und dient als Systemstapelzeiger für die Ausnahmeverarbeitung (z. B. Interrupt). Das 16 Bit breite Statusregister besteht aus einem Systembyte mit Statusbits und einer Interruptmaske sowie dem Anwenderbyte mit den Bedingungsbits für Verzweigungen. Vom Befehlszählregister werden nur die Bits A1...A23 herausgeführt. Damit läßt sich ein Speicherbereich von 16 MByte direkt adressieren.

2.2 Anschlußbelegung

Der Baustein hat 64 Anschlüsse. Adreß- und Datenbus werden getrennt herausgeführt und nicht wie beim 8086 und Z8000 im Multiplexverfahren betrieben. Die Versorgungsspannung beträgt +5 V; die Stromaufnahme max. 1 A.

Der Prozessortakt wird von einem äußeren Taktgeber erzeugt und liegt je nach Prozessorausführung zwischen 4 und 16 MHz. Anders als bei den 8-Bit-Prozessoren der 6800-Serie werden für einen Lesezyklus mindestens vier und für einen Schreibzyklus mindestens fünf Taktzyklen benötigt, so daß die langsamste Ausführung (4 MHz) etwa der 1-MHz-Ausführung des 6800 entspricht.

Die Halt- und Reset-Leitungen müssen beim Einschalten der Versorgungsspannung mindestens 100 ms lang gemeinsam auf Low gehalten werden. Der Halt-Eingang dient wie beim 6800 zum Anhalten des Prozessors. Die Reset-Leitung kann von außen zum Starten des Prozessors verwendet werden; anders als beim 6800 liegen die Reset-Vektoren im unteren Adreßbereich von 000000 aufwärts. Die Reset-Leitung ist gleichzeitig auch ein Ausgang zum Rücksetzen der Peripherie mit einem entsprechenden Befehl.

FC0, FC1 und FC2 zeigen die Art des gerade ausgeführten Zyklus an. IPL0, IPL1 und IPL2 sind Interrupteingänge, mit denen sieben verschiedene Unterbrechungen codiert werden. Auch die Interruptvektoren liegen im unteren Adreßbereich von der Adresse 000000 an aufwärts. BG, BGACK, BR dienen zur Busvergabe bei der Parallelarbeit mit anderen Systemen; der Eingang BERR meldet Busfehler.

Die acht Datenleitungen D0...D7 bilden den unteren Datenweg, D8...D15 den oberen. Sie liegen an getrennten Speicher- und Peripheriebausteinen, da diese byteweise organisiert sind.

Der Ausgang R/W unterscheidet zwischen einem Lese- und einem Schreibzyklus. AS meldet, daß eine gültige Adresse anliegt. Der Ausgang LDS steuert die Datenübertragung auf dem unteren Datenweg (D0...D7), der Ausgang UDS die Übertragung auf dem oberen Datenweg (D8...D15).

Das Adreßbit A0 wird nicht herausgeführt, sondern steuert zusammen mit einem Längencode (Byte, Wort, Doppelwort) in den Befehlen die Datenübertragung auf dem unteren bzw. oberen Datenweg. Die unteren Adreßleitungen liegen an den Adreßeingängen der Speicher- und Peripheriebausteine; aus den oberen Adreßleitungen werden mit einem äußeren Adreßdecoder die Signale zur Bausteinauswahl gebildet.

Man beachte, daß durch das fehlende Adreßbit A0 die Adreßleitung A1 des Prozessors an die Adreßeingänge A0 der Speicher- und Peripheriebausteine angeschlos-

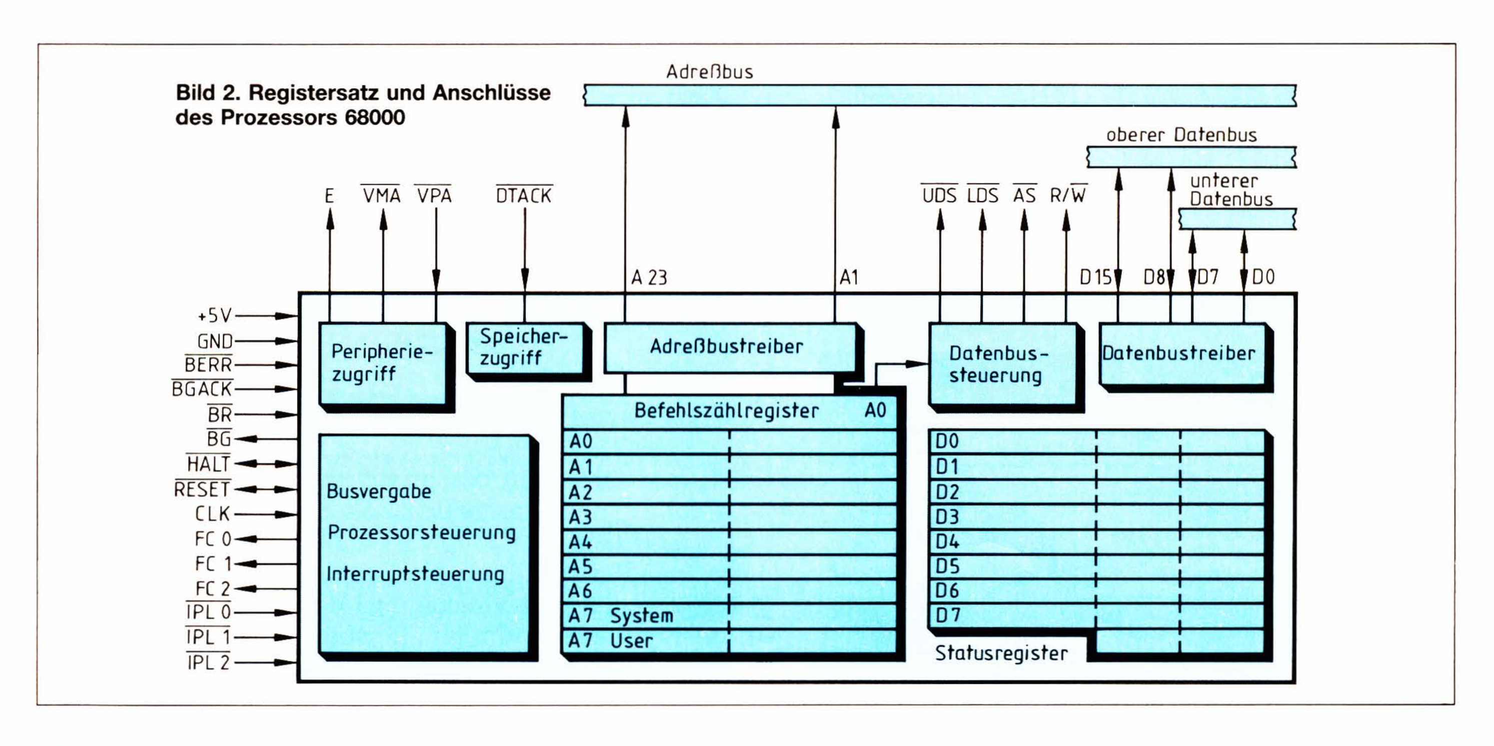


Bild 2. Registersatz und Anschlüsse des Prozessors 68000

sen wird. Diese Verschiebung um ein Bit gilt für alle Adreßleitungen.

2.3 *Adressierung der Speicherbausteine*

Der 68000 kennt den Speicherzugriff auf Bytes (8 Bit), Wörter (16 Bit) und Doppelwörter (32 Bit). Seine Operandenlänge ist im Funktionscode der Befehle enthalten. *Bild 3* zeigt als Modell den Prozessor mit je einem Speicherbaustein am oberen und am unteren Datenweg. Ein Bit unterscheidet zwischen Byte- und Wortzugriff. Die Steuersignale UDS und LDS sind ebenso wie die Auswahleingänge der beiden Speicherbausteine „aktiv Low". UDS = Low adressiert ein Byte aus dem linken Baustein über den oberen Datenweg, LDS = Low adressiert ein Byte aus dem rechten Baustein über den unteren Datenweg.

Bei einem Bytezugriff werden die Bytes von 0, 1, 2, 3, ... an in Einerschritten durchnumeriert. Bytes mit geraden Adressen (0, 2, ...) werden über den oberen Datenweg mit Hilfe des Steuersignals UDS transportiert; Bytes mit ungeraden Adressen (1, 3, ...) über den unteren Datenweg mit dem Steuersignal LDS.

Bei einem Wortzugriff werden die Wörter von 0, 2, 4, ... an in Zweierschritten durchnumeriert. Ungerade Wortadressen sind unzulässig und führen in einen Fehlerstatus. UDS und LDS sind bei einem Wortzugriff gleichzeitig aktiv und transportieren das Wort (= Doppelbyte) gleichzeitig über beide Datenwege.

Doppelwortzugriffe werden in zwei aufeinanderfolgenden Wortzugriffen durchgeführt. Das Doppelwort auf der Adresse 0 besteht aus den beiden Wörtern mit den Adressen 0 und 2 bzw. aus den vier Bytes mit den Adressen 0, 1, 2 und 3.

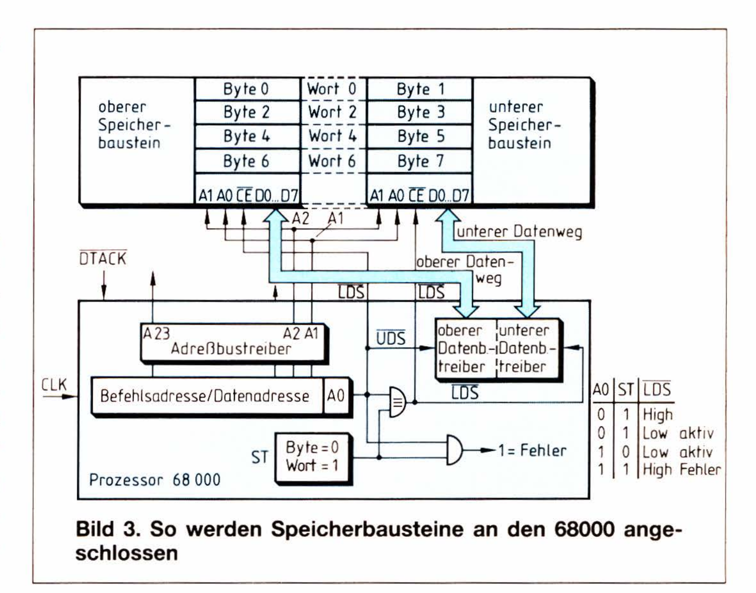


Bild 3. So werden Speicherbausteine an den 68000 angeschlossen

Beim Programmieren der EPROMs z. B. mit dem Monitorprogramm ist darauf zu achten, daß die Befehlswörter und Wortkonstanten getrennt werden. Da alle bekannten EPROM-Bausteine byteweise organisiert sind, sind immer mindestens zwei Bausteine erforderlich. Einer enthält alle Bytes mit gerader Adresse, der andere alle Bytes mit ungerader Adresse.

Der Prozessor 68000 arbeitet bei einem Speicherzugriff asynchron, entsprechend den vereinfachten Zeitdiagrammen *Bild 4* und *Bild 5*. In den beiden ersten Takten eines Lese- bzw. Schreibzyklus macht der Prozessor die Adressen und die Steuersignale R/W, AS, UDS und LDS stabil und prüft anschließend die Eingänge DTACK und VPA. Es werden nun gegebenenfalls so lange Wartetakte eingeschoben, bis die Speicher mit

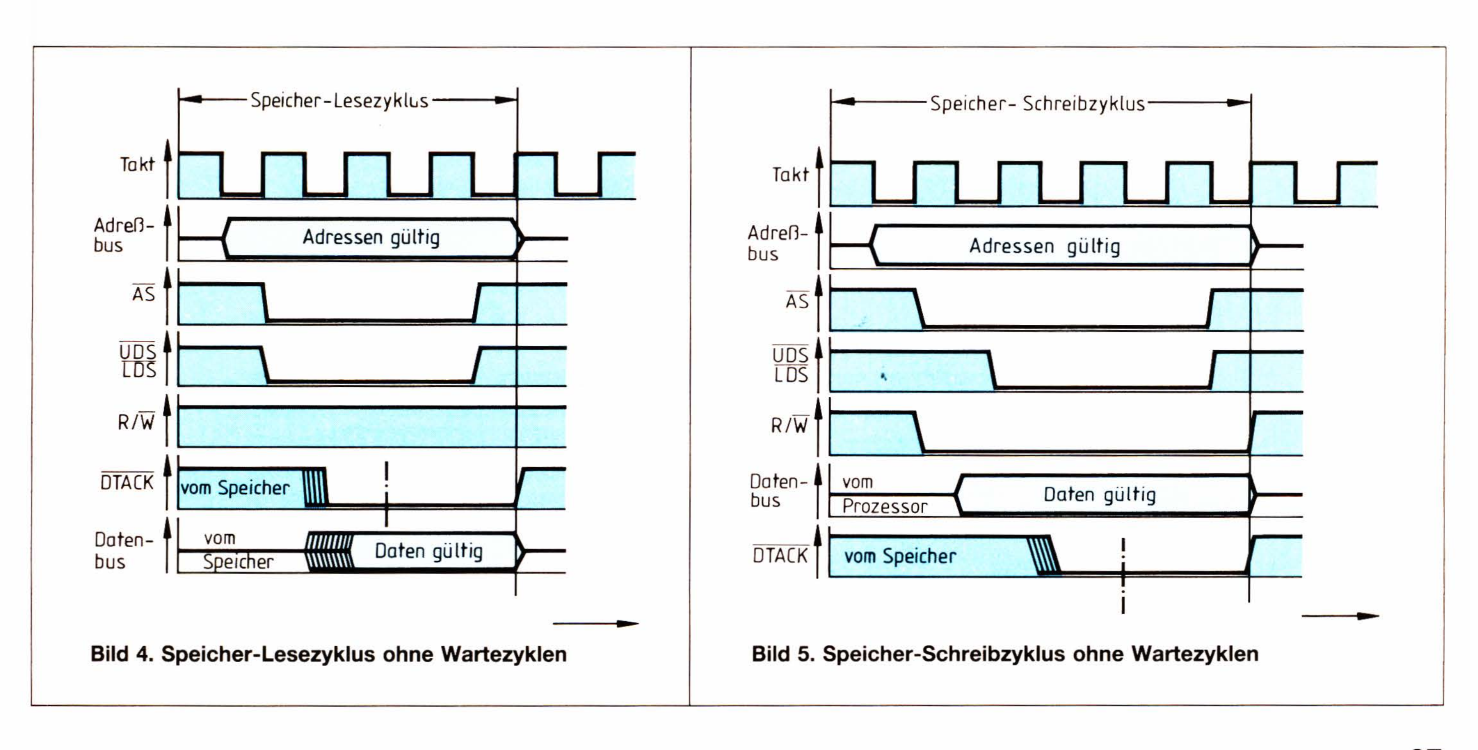


Bild 4. Speicher-Lesezyklus ohne Wartezyklen

Bild 5. Speicher-Schreibzyklus ohne Wartezyklen

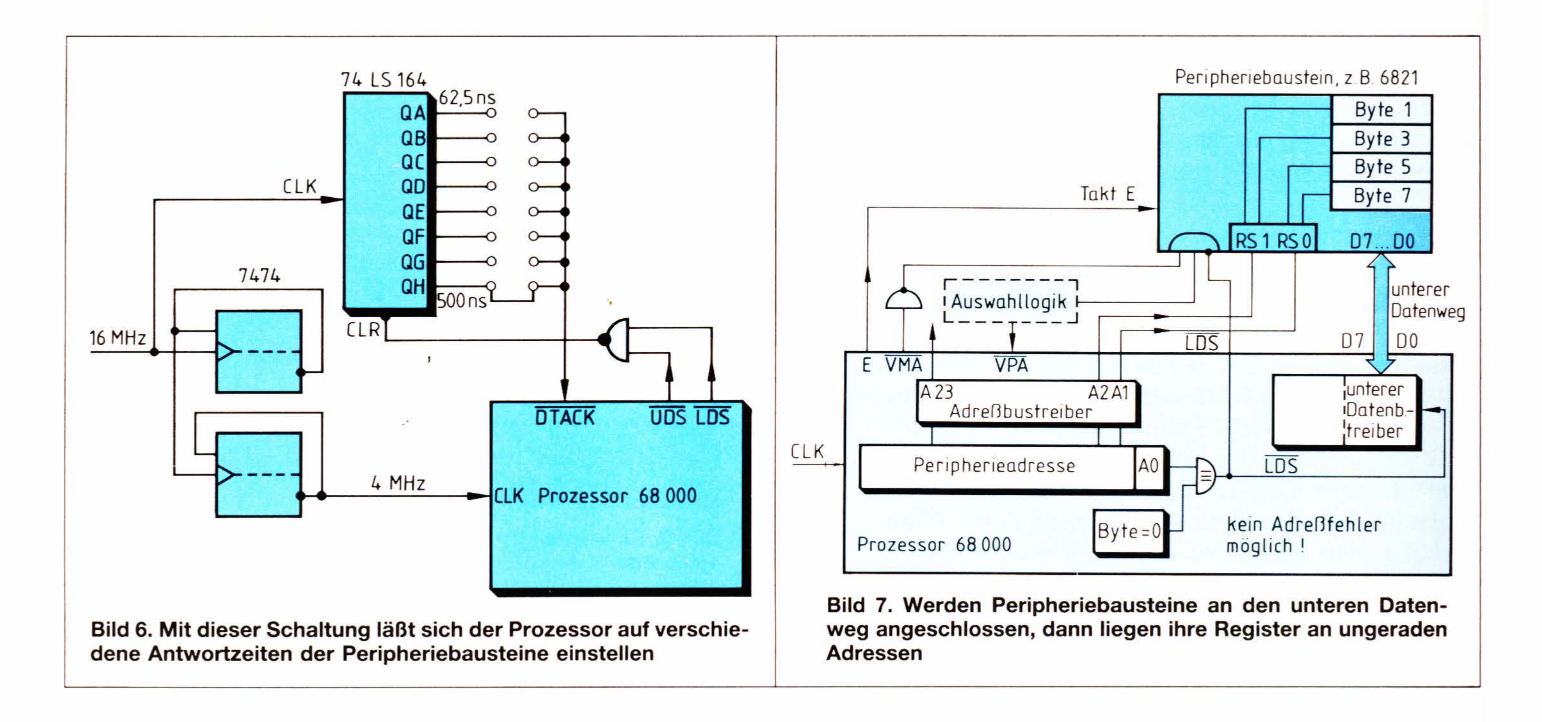


Bild 6. Mit dieser Schaltung läßt sich der Prozessor auf verschiedene Antwortzeiten der Peripheriebausteine einstellen

Bild 7. Werden Peripheriebausteine an den unteren Datenweg angeschlossen, dann liegen ihre Register an ungeraden Adressen

DTACK = Low gemeldet haben, daß sie bereit sind; dann erst setzt der Prozessor den Zyklus mit der Datenübertragung fort. Damit ist es möglich, den Prozessor mit Speichern unterschiedlicher Zugriffszeit zu betreiben. *Bild 6* zeigt eine vom Hersteller vorgeschlagene Schaltung, mit der sich verschiedene Antwortzeiten einstellen lassen. Sie wurde der Systembeschreibung [3] entnommen. In den Zeitdiagrammen *Bild 4* und *Bild 5* fehlen diese Wartetakte; der Prozessor arbeitet mit größtmöglicher Geschwindigkeit.

2.4 Adressierung der Peripherie

Legt man die byteorganisierten Peripheriebausteine wie z. B. die Parallel-Schnittstelle 6821 oder die Serien-Schnittstelle 6850 entsprechend *Bild 7* an den unteren Datenweg, so liegen ihre Register auf ungeraden Adressen. Besondere Peripheriebefehle des Prozessors berücksichtigen die Schrittweite zwei, wenn hintereinanderliegende Register mit Wort- oder Doppelwortbefehlen angesprochen werden sollen.

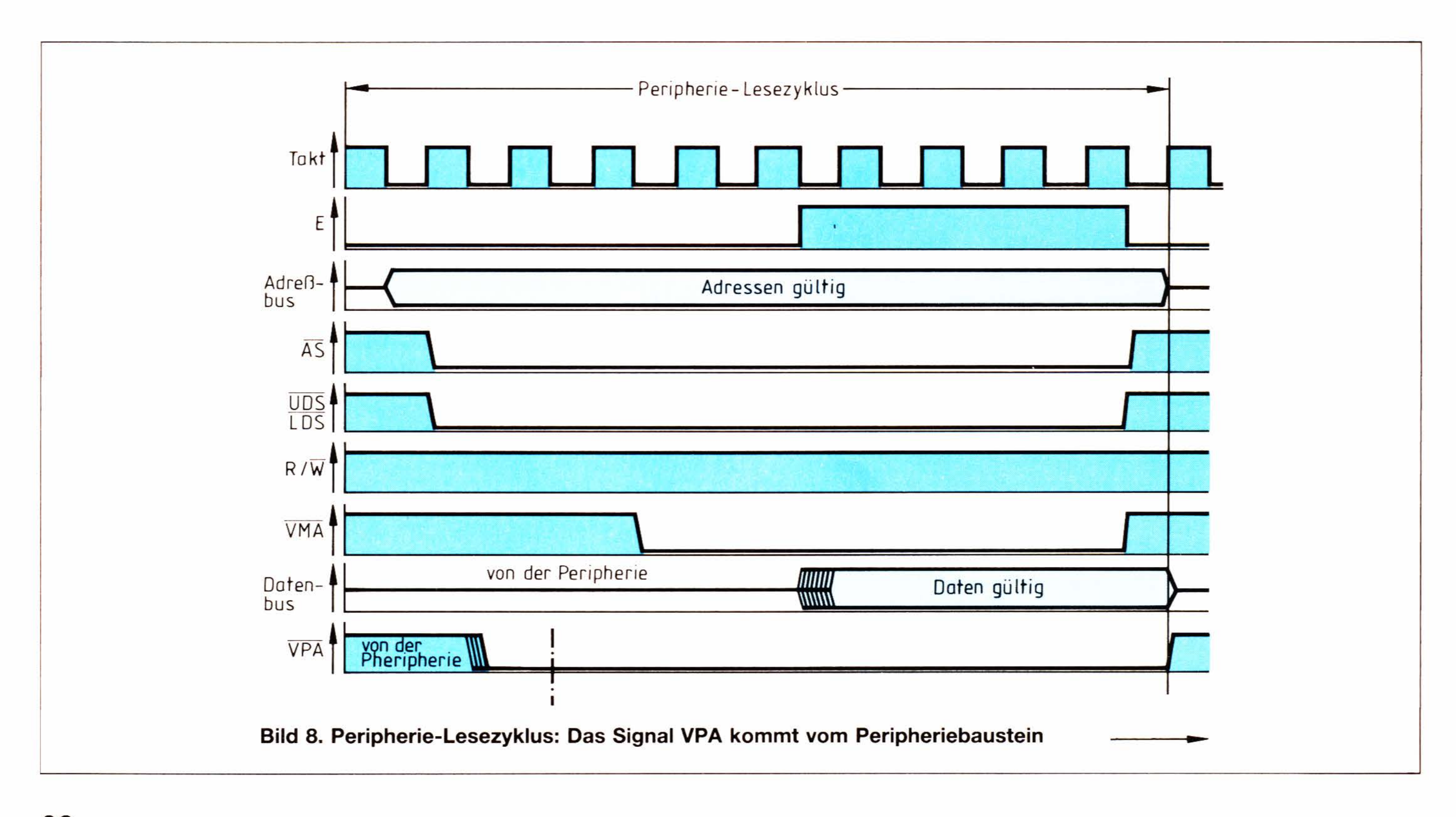


Bild 8. Peripherie-Lesezyklus: Das Signal VPA kommt vom Peripheriebaustein

Wird nach den beiden ersten Takten eines Lese- bzw. Schreibzyklus der VPA-Eingang des Prozessors von einer äußeren Schaltung auf Low gelegt, so beginnt der Prozessor einen Peripheriezugriff entsprechend *Bild 8* und *Bild 9*. Der E-Ausgang des Prozessors liefert eine Taktfrequenz von einem Zehntel des Prozessortaktes, der mit den E-Eingängen der Peripheriebausteine verbunden wird. Bei einem Peripheriezugriff wird der Prozessor gegebenenfalls unter Einfügung von Wartetakten auf das E-Signal synchronisiert, das dem Takt Φ2 des Prozessors 6800 entspricht. Gültige Peripherieadressen werden durch VMA = Low gekennzeichnet; das entsprechende Signal des Prozessors 6800 ist jedoch bei gültigen Adressen High.

2.5 *Ausnahmeverarbeitung*

Als Ausnahme bezeichnet man das Auftreten von

- Reset- und Halt-Signalen,
- Hardwarefehlern wie z. B. Busfehler,
- Softwarefehlern wie z. B. Division durch Null,
- Interruptsignalen an den Eingängen IPL0...IPL2,
- TRAP-Befehlen entsprechend dem SWI-Befehl des 6800 und
- Einzelschrittanforderungen durch das T-Bit des Statusregisters.

Alle Ausnahmen bewirken einen Abbruch des laufenden Programms und den Start eines entsprechenden Ausnahmeprogramms, dessen Startadresse als Vektor im Bereich von \$0000...\$03FF abzulegen ist.

Beim Auftreten eines Interrupts wird das auslösende Interruptmuster wieder auf den Adreßleitungen A1...A3 ausgegeben; die restlichen Adreßleitungen werden auf High gelegt. Es gibt nun zwei verschiedene Möglichkeiten, die Startadresse des Interruptprogramms zu bestimmen:

Meldet sich aufgrund der ausgesendeten Adresse ein Speicher mit VPA = High und DTACK = Low, so wird die am Datenbus anliegende Vektornummer in die Startadresse eines Interruptprogramms umgesetzt. Die Vektortabelle liegt im Bereich von \$0000...\$03FF.

Wird bei einem Interrupt-Zyklus dagegen VPA = Low erkannt, so führt der Prozessor einen Peripherie-Lesezyklus durch und entnimmt die Startadresse des Interruptprogramms einer Tabelle, die im Bereich von \$0064...\$007F angeordnet ist. Wegen des gleichzeitig durchgeführten Peripherie-Lesezyklus, in dem auch das VMA-Signal aktiv ist, muß dafür gesorgt werden, daß die Peripherie bei einem derartigen Interrupt gesperrt wird. Sonst könnten Interrupt-Anzeigen versehentlich zurückgesetzt werden.

Der Anwender kann also durch entsprechende Adreßdecodierung für den Fall einer Interruptanforderung zwischen den beiden Interruptmöglichkeiten wählen.

3 Schaltungsdetails

Da es sich lediglich um ein einfaches Übungsgerät handelt, wurde die langsamste Prozessorausführung von 4 MHz gewählt. Das DTACK-Signal, mit dem die Speicher ihre Bereitschaft zur Datenübertragung melden, wird vom Prozessorausgang AS abgeleitet, so daß Wartetakte entfallen.

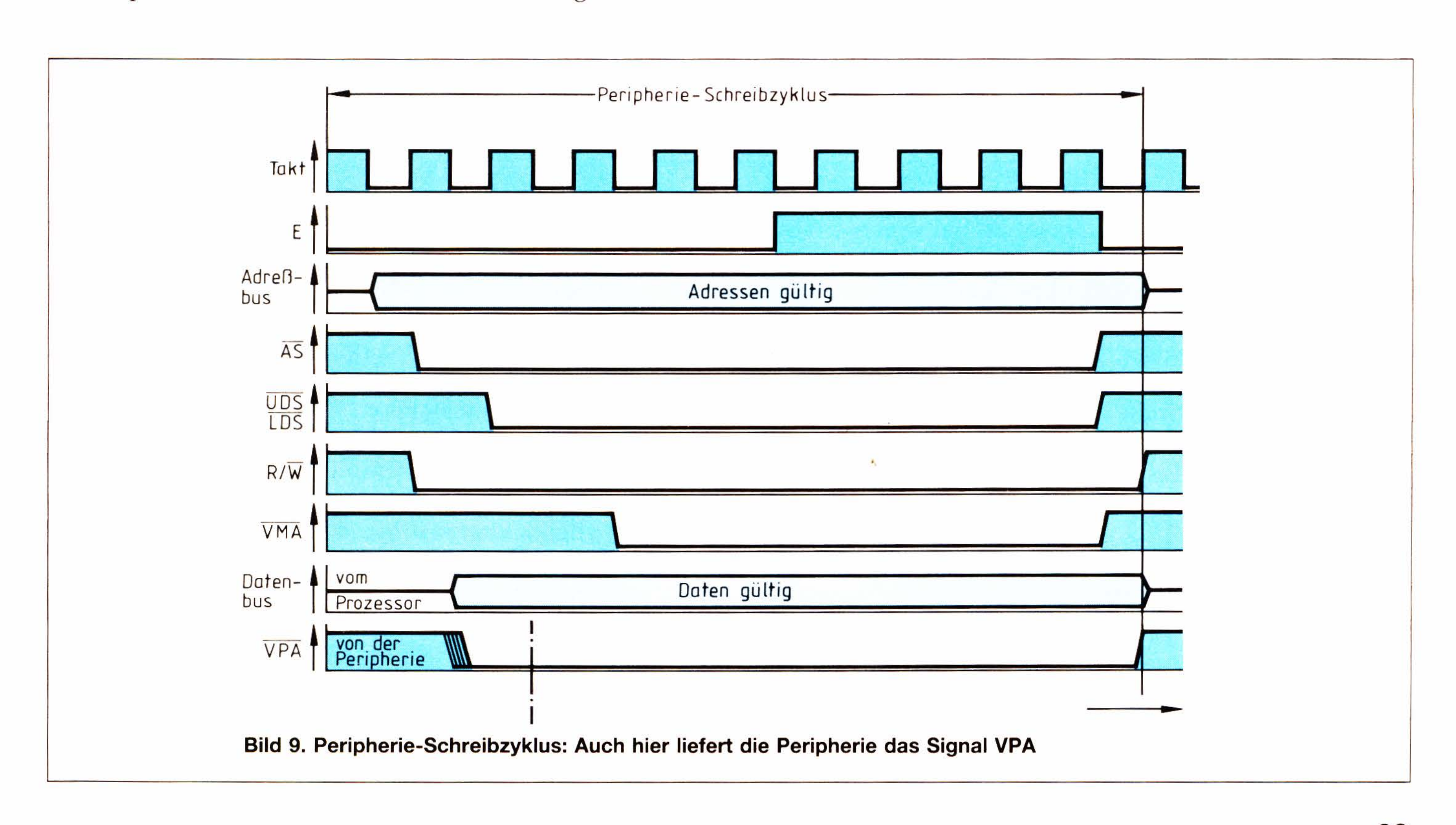


Bild 9. Peripherie-Schreibzyklus: Auch hier liefert die Peripherie das Signal VPA

Adreßdecodierung des 68000-Systems

Baustein-Adr.			frei		Decod.		Bausteine									
			A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	
68000-Adresse			A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	L/U
2716	0FFF	Y0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
	:		:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:
	0000		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2114	27FF	Y2	0	0	1	0	x	1	1	1	1	1	1	1	1	1	1	1
	:		:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:
	2000		0	0	1	0	x	0	0	0	0	0	0	0	0	0	0	0
6850	3403	LY3	0	0	1	1	0	1	x	x	x	x	x	x	x	x	1	1
	3401		0	0	1	1	0	1	x	x	x	x	x	x	x	x	0	1
6821	3807	LY3	0	0	1	1	1	0	x	x	x	x	x	x	x	1	1	1
	3805		0	0	1	1	1	0	x	x	x	x	x	x	x	1	0	1
	3803		0	0	1	1	1	0	x	x	x	x	x	x	x	0	1	1
	3801		0	0	1	1	1	0	x	x	x	x	x	x	x	0	0	1
Int.	$\overline{VPA}$= Low	LY3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Die *Tabelle* zeigt den Adreßplan; *Bild 10* den Schaltplan. Zur Bausteinauswahl dient ein 74LS155, der zwei 1-aus-4-Decodierer mit gemeinsamen Adreßeingängen A12 und A13 enthält. Einer der beiden Decodierer steuert zusammen mit dem Signal UDS die am oberen Datenweg liegenden Speicherbausteine, der andere mit dem Signal LDS die am unteren Datenweg liegenden Speicher- und Peripheriebausteine. Die freien Adreßleitungen des Systems wurden bei der Ermittlung der hexadezimalen Adressen auf Null gesetzt.

Die beiden EPROMs 2716 bilden einen 4 KByte großen Festwertspeicher für ein Monitorprogramm auf den Byteadressen $0000....$0FFF. Dieser Bereich enthält die Vektortabellen für Reset und die Ausnahmeverarbeitung. Beide Bausteine liegen an den Ausgängen Y0 des Decodierers. Ihre OE-Eingänge wurden auf Low gelegt.

Die vier RAM-Bausteine 2114 bilden einen 2 KByte großen Arbeitsspeicher auf den Byteadressen $2000...$27FF. Davon belegt der Monitor etwa 128 Bytes, der Rest steht dem Benutzer zur Verfügung. Die Bausteine liegen an den Decodiererausgängen Y2.

Der Ausgang Y3 des mit LDS freigegebenen Decodierers steuert die Peripherie und bildet gleichzeitig das VPA-Signal, mit dem ein Peripherie-Zyklus eingeleitet wird. Da bei einer Interruptanforderung A12 und A13 ebenfalls beide High werden, wird in diesem Fall VPA = Low und leitet einen Peripherie-Lesezyklus ein. Dabei wird ein Interruptprogramm gestartet, dessen Startadresse auf einer der sieben Adressen zwischen $0064 und $007C liegt. Innerhalb des Peripheriebereiches dienen die Adreßleitungen A11 und A12 zur linearen Auswahl der beiden Peripheriebausteine, die beide nur für VMA = Low freigegeben werden.

Die vier Register der Parallel-Schnittstelle 6821 liegen auf den Adressen $3801, $3803, $3805 und $3807. Durch die Auswahlbedingung A10 = Low wird die Schnittstelle bei einem Interrupt gesperrt.

Die beiden Register der Serien-Schnittstelle 6850 liegen auf den Adressen $3401 und $3403. Durch die Auswahlbedingung A11 = Low wird der Baustein bei einem Interrupt gesperrt.

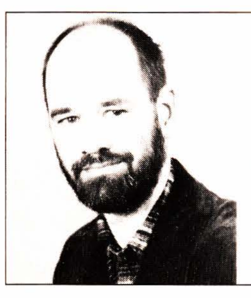
Der Decodiererausgang Y3, der von UDS gesteuert wird, und die beiden Decodiererausgänge Y1 sind für Speichererweiterungen frei.

Das RC-Glied am Ausgang des Entprell-Flipflops der Reset-Taste soll beim Einschalten der Versorgungsspannung die Reset- und Halt-Eingänge für etwa 100 ms auf Low halten, um einen sicheren Anlauf des Prozessors zu ermöglichen.

Die Abort-Taste wird entprellt auf den IPL2-Eingang geführt und dient zum Abbruch von Benutzerprogrammen. Über die beiden restlichen Interrupteingänge kann der Benutzer frei verfügen, sie werden vom Monitor nicht benutzt.

Die Bus-Steuereingänge BGACK und BR sowie BERR werden konstant auf High gelegt und nicht verwendet. Die Ausgänge BG sowie FC0...FC2 bleiben offen und unbenutzt.

Ein 4-MHz-Quarz steuert einen einfachen Taktgeber für den CLK-Eingang des Prozessors. Der Peripherie-



Prof. Dipl.-Ing. Günter Schmitt ist gebürtiger Berliner. Er studierte an der dortigen TU und arbeitete von 1963 bis 1969 als Entwicklungsingenieur bei Siemens in Berlin. Seit 1969 lehrt er an der Fachhochschule der Deutschen Bundespost Dieburg in den Fächern Mikrocomputertechnik, Datenverarbeitung und Meßtechnik. Im Wintersemester 1980/81 arbeitete er in einem Praxissemester bei der Dr. Weiß GmbH in Schriesheim. Hobby: Reiten auf Fjordpferden

takt E wird vom Prozessor erzeugt und den beiden Schnittstellen zugeführt.

4 Mechanischer Aufbau

Die Schaltung wurde auf einer Europakarte in Fädeltechnik aufgebaut. Für den Prozessor wurden anstelle eines 64poligen Sockels acht 16polige Sockel aneinandergereiht. Auf der mit Lötaugen und einem Anschluß für einen 64poligen Stecker versehenen Karte wurden zunächst die Sockel befestigt und die Stromversorgungsleitungen mit 0,8 mm starkem isoliertem Draht gezogen. Alle anderen Verbindungen bestehen aus 0,2 mm starkem Kupferlackdraht, der um die Stifte gewickelt und anschließend verlötet wurde. Die Leitungsführung erfolgte auf dem kürzesten Weg. Eine in ca. 5 mm Abstand befestigte kupferkaschierte zweite Karte schützt die Verdrahtungsseite.

Eine Stirnseite der Karte trägt eine 25polige Buchse für die Leitungen zwischen der Serien-Schnittstelle und dem Terminal sowie die beiden Taster für Reset und Abort. Die andere Seite der Karte enthält einen 64poligen Stecker mit den Stromversorgungsanschlüssen (+5 V, +12 V, −12 V und Masse), den Leitungen für die serielle Datenübertragung, den 20 Ein-/Ausgängen der Parallel-Schnittstelle und den beiden Leitungen für Reset und Abort. Die Karte kann auf ein Grundgerät aufgesteckt werden, das die Stromversorgung, eine neunstellige Siebensegment-Anzeigeeinheit und ein Tastenfeld enthält. Dieses Grundgerät wird auch für ähnliche mit anderen Prozessoren arbeitende Karten verwendet.

Der Verfasser dankt Herrn *Dr. Weiß* von der Dr. Weiß GmbH Schriesheim für die freundliche Unterstützung im Praxissemester sowie den Herren *Dipl.-Ing. G. Stumpf* und *Dipl.-Ing. G. Winter*, die in ihrer Abschlußarbeit an der FH Dieburg ein eigenes 68000-System aufgebaut haben.

Bibliografische Notizen sind erschienen in der ELEKTRONIK 1978, H. 5.

Literatur

[1] 16-Bit Mikroprozessor Benützer-Handbuch. Firmenschrift MC68000UM(AD), Motorola.
[2] 16-Bit Microprocessing Unit. Datenblatt, Motorola.
[3] Design Module User's Guide. Firmenschrift MEX68KDM(D2), Motorola.
[4] *Daniels, G., Lösel, M.*: Ein 16-Bit-Mikroprozessor in HMOS-Technik. ELEKTRONIK 1979, H. 10, S. 47...53.
[5] *Gößler, R.*: 16-Bit-Mikroprozessoren der dritten Generation. ELEKTRONIK 1980, H. 22, S. 61...73.
[6] *Männer, R., Deluigi, B.*: 16-Bit-Prozessoren im Vergleich. 1. Teil: ELEKTRONIK 1981, H. 5, S. 77...83. 2. Teil: ELEKTRONIK 1981, H. 6, S. 119...124. 3. Teil: ELEKTRONIK 1981, H. 7, S. 101...107.

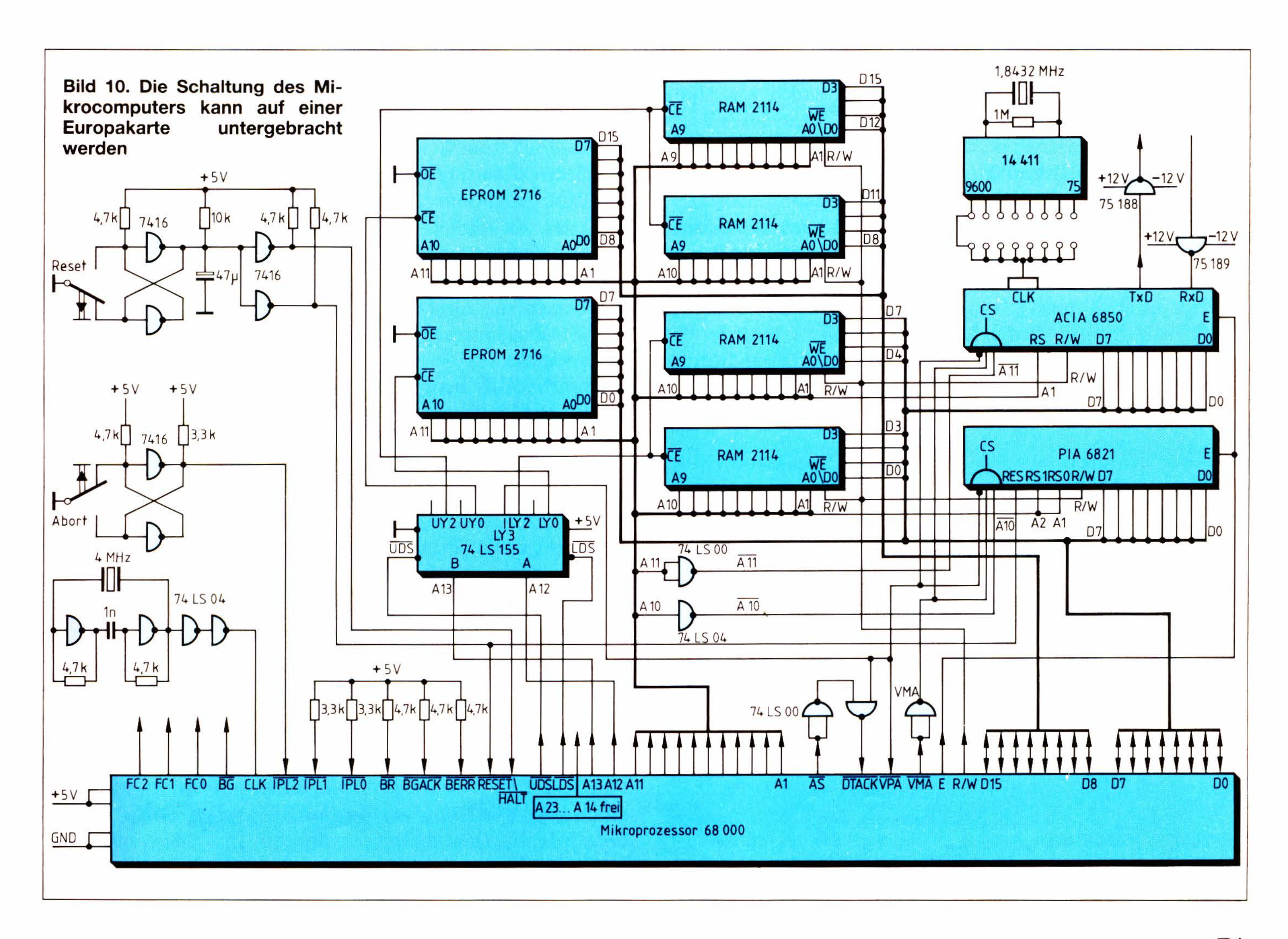


Bild 10. Die Schaltung des Mikrocomputers kann auf einer Europakarte untergebracht werden

Prof. Dipl.-Ing. Günter Schmitt

Einfaches Monitorprogramm bedient Ein-/Ausgabe-Terminal

Der ab Seite 65 vorgestellte Mikrocomputer enthält eine Serien-Schnittstelle mit V.24-Treibern zum Anschluß eines Ein-/Ausgabe-Terminals. Mit dem vorgestellten Monitorprogramm kann der Benutzer über das Terminal Speicherstellen anzeigen und ändern sowie Programme starten und testen.

1 Die Hardware

Bild 1 zeigt einen Überblick über die vom Monitor benutzte Hardware. Zwei EPROM-Bausteine 2716 auf den Byteadressen $0000...$0FFF bilden einen 4-KByte-Festwertspeicher mit den Ausnahmevektoren, dem Monitorprogramm und den Tabellen.

Vier RAM-Bausteine 2114 auf den Byteadressen $2000...$27FF bilden den Arbeitsspeicher für den Benutzer und für den Monitor. Eine Serien-Schnittstelle 6850 mit der Adresse $3401 für das Steuerregister und $3403 für das Datenregister dient zur seriellen Datenübertragung. Die Übertragungsgeschwindigkeit (Baudrate) kann an einem Frequenzteiler zwischen 9600 und 75 Bd eingestellt werden. Als Terminal wurde ein Alphatronic-Tischrechner verwendet, der eine Serien-Schnittstelle mit dem Baustein 8251A enthält.

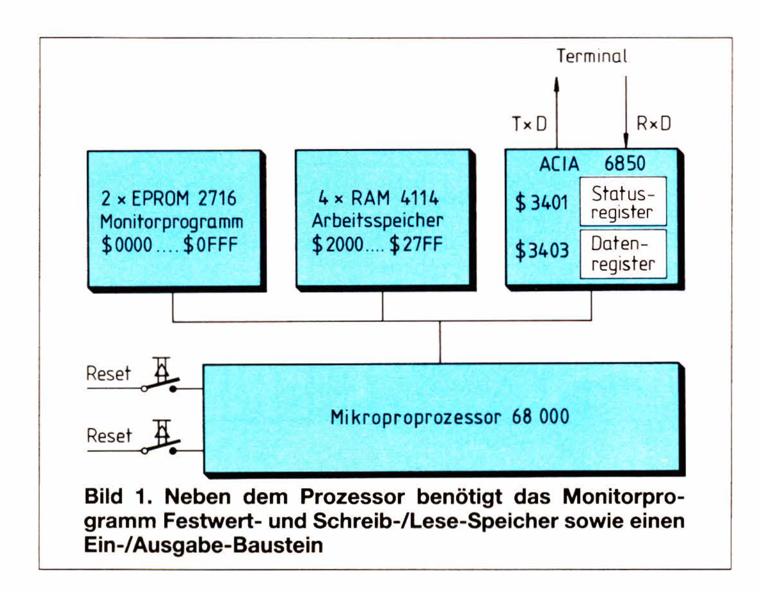


Bild 1. Neben dem Prozessor benötigt das Monitorprogramm Festwert- und Schreib-/Lese-Speicher sowie einen Ein-/Ausgabe-Baustein

Die Reset-Taste dient zum Starten des Monitorprogramms; mit der Abort-Taste kann der Benutzer sein Programm abbrechen. Für die Einzelschrittsteuerung wird keine Hardware, sondern das T-Bit des Statusregisters verwendet.

2 Monitorfunktionen

Tabelle 1 faßt die Kommandos und Funktionen des Monitors zusammen: Nach einem Reset meldet sich der Monitor mit dem Zeichen „›" am linken Rand und erwartet die Eingabe des Benutzers.

Alle einzugebenden Werte (Adressen und Inhalte) sind Hexadezimalzahlen ohne das in der Assemblerschreibweise übliche $-Zeichen. Die Eingabe wurde auf vier Stellen beschränkt. Dies entspricht zwei Bytes oder einem Wort.

Kommandos bestehen aus einer Adresse und einem Kommandozeichen oder einem Kommandozeichen allein. Kommandozeichen können alle Zeichen außer den Ziffern 0...9 und den Buchstaben A...Z sein.

Funktionsnamen bestehen aus vier Buchstaben oder Ziffern. Um sie von einer hexadezimalen Adresse unterscheiden zu können, muß mindestens ein Zeichen von den hexadezimalen Ziffern 0...9 und A...F verschieden sein. Die Eingabe eines Funktionsnamens wird durch ein Leerzeichen abgeschlossen. Funktionen fordern meist den Benutzer auf, Steuergrößen wie z. B. Adressen einzugeben.

Da bei der Eingabe von Adressen und Funktionsnamen häufig Fehler auftreten können, die der Benutzer sofort bemerkt, wertet der Monitor immer nur die letzten vier Zeichen vor dem abschließenden Kommandozeichen aus; der Benutzer kann also seinen Fehler sofort berichtigen. Das Abbruchzeichen Control-X mit dem ASCII-Code $18 bricht jede Eingabe ab und führt in die Grundstellung des Monitors zurück.

```
>
> FILL  ANF>2000  END>2100  WRT>0000 (CR)
> 2000  0000-7000
* 2002 0000-5280
* 2004 0000-4EF8
* 2006 0000-2002 (CR)
> 2000;G
* ABRT  PC= 00002002  SR= 2000  SP= 00002780
    D0= 00180FB7  D1= 00000000  D2= 00000000  D3= 00000000
    D4= 00000000  D5= 00000000  D6= 00000000  D7= 00000000
    A0= 00000000  A1= 00000000  A2= 00000000  A3= 00000000
    A4= 00000000  A5= 00000000  A6= 00000000  A7= 00002780
```

Bild 2. Eingabe, Start und Abbruch eines Testprogramms. CR bedeutet, daß an dieser Stelle die Carriage-Return-Taste betätigt wird

Bild 2 zeigt als Beispiel die Eingabe, den Start und den Abbruch eines Testprogramms, das zunächst das Register D0 löscht und dann in einer unendlichen Schleife laufend um 1 erhöht.

2.1 Ausgeben von Speicherinhalten

Hinter dem Eingabezeichen „›" gibt der Benutzer die vierstellige hexadezimale Adresse des auszugebenden Wortes ein und schließt sie mit dem Kommandozeichen „Leerzeichen" ab. Der Monitor gibt den Inhalt des gewünschten Wortes gefolgt von einem Bindestrich aus und erwartet nun eine Eingabe des Benutzers:

- Ein Zeilenvorschub oder Leerzeichen bewirkt, daß der Monitor die Adresse um zwei erhöht und damit den Inhalt des folgenden Wortes auf der nächsten Zeile anzeigt.
- Ein Minuszeichen bewirkt, daß die Adresse um zwei vermindert und damit der Inhalt des vorhergehenden Wortes auf der nächsten Zeile angezeigt wird.
- Ein Wagenrücklauf bricht das Kommando ab; der Monitor kehrt in die Grundstellung zurück.

Für die Ausgabe eines größeren Speicherbereiches steht die Dump-Funktion zur Verfügung. Sie fordert den Benutzer auf, Anfangs- und Endadresse des gewünschten Bereiches einzugeben. Die Adressen werden so verändert, daß immer acht Wörter auf einer Ausgabezeile stehen. Am linken Rand erscheint die achtstellige Adresse des ersten Wortes einer Zeile.

2.2 Eingeben von Speicherinhalten

Der Benutzer gibt zunächst wie bei der Anzeige die Adresse des gewünschten Wortes ein, und der Monitor zeigt den alten Inhalt an.

Der neue Wert wird hinter dem Bindestrich eingegeben und durch eins der folgenden Kommandozeichen abgeschlossen:

- Ein Zeilenvorschub oder Leerzeichen setzt das Kommando mit dem folgenden Wort fort.
- Ein Minuszeichen setzt das Kommando mit dem vorhergehenden Wort fort.
- Ein Wagenrücklauf beendet das Kommando.

Die FILL-Funktion speichert ein konstantes Wort in einen größeren Speicherbereich. Die Funktion fordert den Benutzer auf, die Anfangsadresse, die Endadresse und die Wortkonstante einzugeben.

2.3 Starten eines Programms

Der Benutzer gibt die Startadresse des Programms gefolgt von den Kommandozeichen „;G" ein, und der Monitor gibt die Steuerung an das Benutzerprogramm ab.

Der Benutzer kann ein Programm auch beim augenblicklichen Stand des Befehlszählers allein mit dem Kommandozeichen „;" starten. Diese Möglichkeit wird besonders dann verwendet, wenn das Programm angehalten wurde und nun fortgesetzt werden soll.

Neben dem reinen Start des Programms führt der Monitor eine Reihe von Operationen durch, die erst im Abschnitt 2.5 über die Testhilfen erklärt werden. Dazu gehören:

- Laden der Benutzerregister,
- Setzen des Haltepunktes,
- Freigabe des Interrupts für die Abort-Taste und
- Setzen des T-Bits für die Programmverfolgung.

2.4 Anhalten eines Programms

Der Benutzer kann mit einem Sprungbefehl von seinem Programm aus in den Monitor zurückspringen. Der beste Einsprungpunkt ist die Adresse $02DC, die in die Grundstellung des Monitors zurückführt. Auf der Ausgabe erscheint das Eingabezeichen „›" als Meldung des Monitors in der Grundstellung.

Tabelle 1. Kommandos und Funktionen des Monitors

Eingabe	Bedeutung
› adresse BL	Wort anzeigen und neu eingeben
› zeile BL	adresse + 2
› zeile LF	adresse + 2
› zeile –	adresse – 2
› zeile CR	Ende des Kommandos
› adresse;G	Programm starten ab Adresse
› ;	Programm starten ab Befehlszähler
› adresse;V	Haltepunkt setzen bei Adresse
› :	Haltepunkt löschen
› adresse;T	Einzelschritt ab Adresse
› /	Einzelschritt ab Befehlszähler
› .	Ausgeben aller Register
› DUMP BL	Speicherbereich ausgeben
› FILL BL	Speicherbereich mit Wort füllen

Tabelle 2. Registeradressen im RAM-Bereich

Register	Adresse 1. Wort	Adresse 2. Wort	Bedeutung
D0	$27AA	$27AC	Datenregister
D1	$27AE	$27B0	Datenregister
D2	$27B2	$27B4	Datenregister
D3	$27B6	$27B8	Datenregister
D4	$27BA	$27BC	Datenregister
D5	$27BE	$27C0	Datenregister
D6	$27C2	$27C4	Datenregister
D7	$27C6	$27C8	Datenregister
A0	$27CA	$27CC	Adreßregister
A1	$27CE	$27D0	Adreßregister
A2	$27D2	$27D4	Adreßregister
A3	$27D6	$27D8	Adreßregister
A4	$27DA	$27DC	Adreßregister
A5	$27DE	$27E0	Adreßregister
A6	$27E2	$27E4	Adreßregister
A7	$27E6	$27E8	User-Stapelzeiger
SP	$27A6	$27A8	System-Stapelzeiger
SR	$27A4	—	Statusregister

Ein Benutzerprogramm kann durch die Abort-Taste mit einem Interrupt abgebrochen werden. Auf der Ausgabe erscheint die Meldung ABRT zusammen mit dem Inhalt des Befehlszählregisters, des Statusregisters und des System-Stapelzeigers.

Bricht der Benutzer sein Programm mit der Reset-Taste ab, so wird der Monitor neu gestartet. Dabei werden Befehlszähler, Statusregister und beide Stapelzeiger des Benutzers auf konstante Anfangswerte gesetzt; alle anderen Register werden gelöscht.

Für Testzwecke kann ein Programm auch an einem Haltepunkt angehalten oder im Einzelschritt verfolgt werden.

2.5 Testen eines Benutzerprogramms

Die Programmierung dieser Testhilfen ist sehr aufwendig. Es können daher an dieser Stelle nur einfache Beispiele gezeigt werden:

Haltepunkte sind ein bevorzugtes Mittel, um die Wirkung von Programmverzweigungen zu überprüfen, da das Benutzerprogramm an dieser Stelle angehalten wird.

Der Benutzer kann einen Befehl seines Programms mit einem Haltepunkt markieren. Dazu gibt er die Adresse gefolgt von den Kommandozeichen „;V" für Verfolgungspunkt ein. Der Monitor trägt diese Adresse zunächst in eine Tabelle ein. Der Benutzer kann durch Eingabe des Kommandozeichens „:" den Haltepunkt wieder löschen. Beim Start des Programms wird der auf dieser Adresse liegende Funktionscode gegen den Code des Trap-Befehls ausgetauscht. Dieser entspricht dem Befehl SWI des Prozessors 6800 und bewirkt eine Ausnahmeverarbeitung, die in den Monitor zurückführt. Da der alte Code wieder zurückgeschrieben wird, merkt der Benutzer unter normalen Umständen nichts davon. Bei dem Erreichen eines Haltepunktes gibt der Monitor die Meldung VSTP zusammen mit dem Inhalt des Befehlszählers, des Statusregisters und des System-Stapelzeigers aus.

Beim Setzen eines Haltepunktes sind folgende Besonderheiten zu beachten:

- Auf die Adresse $0000 kann kein Haltepunkt gesetzt werden.
- Der Haltepunkt muß im RAM-Bereich liegen.
- Der Haltepunkt muß an einem Funktionscode liegen.
- Das Programm darf nicht mit Reset abgebrochen werden, da in diesem Fall der Monitor den alten Code nicht zurückschreiben kann und der TRAP-Befehl mit dem Code $4E4F im Programm verbleibt.
- Beim Erreichen eines Haltepunktes wurde der Befehl, an dem der Haltepunkt gesetzt wurde und auf den der Befehlszähler zeigt, noch nicht ausgeführt.

Der vorliegende Monitor kennt nur einen Haltepunkt, der nur einen der 16 Trap-Befehle verwendet. Durch eine Erweiterung der Haltepunktliste und bei Verwendung weiterer Trap-Befehle können die Möglichkeiten des Monitors wesentlich erweitert werden.

Im Einzelschrittbetrieb will der Benutzer nur einen Befehl seines Programms ausführen und anschließend in den Monitor zurückkehren, um z. B. den Inhalt von Registern oder Speicherstellen zu überprüfen. Durch Eingabe einer Startadresse gefolgt von den Kommandozeichen „;T" führt der Monitor nur diesen einen Befehl aus und gibt dann die Meldung TRAC für TRACE und den Stand des Befehlszählregisters, des Statusregisters und des System-Stapelzeigers aus. Durch Eingabe des Kommandozeichens „/" allein kann ohne Eingabe einer Adresse der Befehl beim augenblicklichen Stand des Befehlszählers ausgeführt werden. Der Befehl, auf den der Befehlszähler nach einem Einzelschritt zeigt, wurde noch nicht ausgeführt.

Beim Prozessor 68000 ist bereits in der Ablaufsteuerung der Einzelschrittbetrieb vorgesehen. Vor dem Start setzt der Monitor das T-Bit des Benutzer-Statusregisters, so daß der Prozessor automatisch nur einen Befehl ausführt und dann eine Trace-Ausnahmeverarbeitung einleitet, die in den Monitor zurückführt. Von der zusätzlichen Möglichkeit, mit Hilfe der Halt-Leitung einen Hardware-Einzelschritt auszuführen, wurde kein Gebrauch gemacht. Bei Prozessoren, die keine eingebaute Einzelschrittsteuerung enthalten, startet man mit dem Benutzerprogramm zugleich einen äußeren Zähler oder Timer, der im ersten Zyklus des auszuführenden Befehls einen Interrupt auslöst und damit das Programm abbricht.

Während des Programmtests kann der Benutzer mit Hilfe des Monitors Speicherstellen prüfen und verändern. Die Register des Prozessors, mit denen der Benutzer arbeitet, werden vom Monitor in einem besonderen Speicherbereich verwaltet. Sie werden bei jedem Start eines Benutzerprogramms geladen und bei jedem Programmabbruch außer bei einem Reset gerettet. Ein Reset setzt Befehlszähler, Statusregister und die beiden Sta-

pelzeiger auf einen Anfangswert und löscht alle anderen Register. Das Kommandozeichen „." bewirkt eine Ausgabe des gesamten Registersatzes.

Die Registerverwaltung wurde in dem vorliegenden Monitorprogramm auf die Ausgabe des gesamten Registersatzes beschränkt. *Tabelle 2* zeigt die Adressen der Register im RAM-Bereich für den Fall, daß der Benutzer Registerinhalte mit Hilfe des Monitors verändern will. Man beachte, daß alle Register mit Ausnahme des Statusregisters 32 Bit breit und daher in einem Doppelwort abgespeichert sind. Da der Monitor nur den Zugriff auf Wörter zuläßt, muß der Benutzer Registerinhalte mit zwei aufeinanderfolgenden Kommandos verändern.

2.6 Fehlermeldungen

Der Monitor unterscheidet Fehler während der Eingabe des Benutzers und Fehler, die während der Ausführung eines Benutzerprogramms auftreten. Die Behandlung beider Fehlermöglichkeiten wurde auf ein Minimum beschränkt. *Tabelle 3* faßt die Fehlermeldungen zusammen.

3 Programmstruktur

Das Struktogramm (*Bild 3*) zeigt eine Übersicht über den Aufbau des Monitorprogramms.

Nach dem Setzen von Anfangswerten geht der Monitor in die Grundstellung und erwartet hinter dem Zeichen „>" eine Eingabe des Benutzers. An dieser Stelle liegt das wichtigste Programmstück, das die Verteilung auf die einzelnen Kommandos und Funktionen vornimmt. Es arbeitet mit einem Eingabe-Unterprogramm zusammen, das so lange Zeichen aufnimmt, bis ein Kommandozeichen erscheint, also ein Zeichen außer einer Ziffer 0...9 oder einem Buchstaben A...Z. Es können drei Fälle auftreten:

- Wird als erstes Zeichen ein Kommandozeichen eingegeben, so handelt es sich um ein Kommando ohne Adresse.

Tabelle 3. Fehlermeldungen des Monitors

Meldung	Fehlerursache
HEX?	Nicht-hexadezimaler Wert
RAM?	Kein RAM-Bereich
COM?	Kommando nicht vorhanden
FUN?	Funktion nicht vorhanden
* ERR:0	Busfehler
* ERR:1	Adreßfehler
* ERR:2	Operationsfehler
* ERR:3	Division durch Null
* ERR:4	CHK-Befehl
* ERR-5	TRAP-V-Befehl
* ERR:6	Statusverletzung
* ERR:7	1010-Emulator
* ERR:8	1111-Emulator
* ERR:9	Interruptfehler

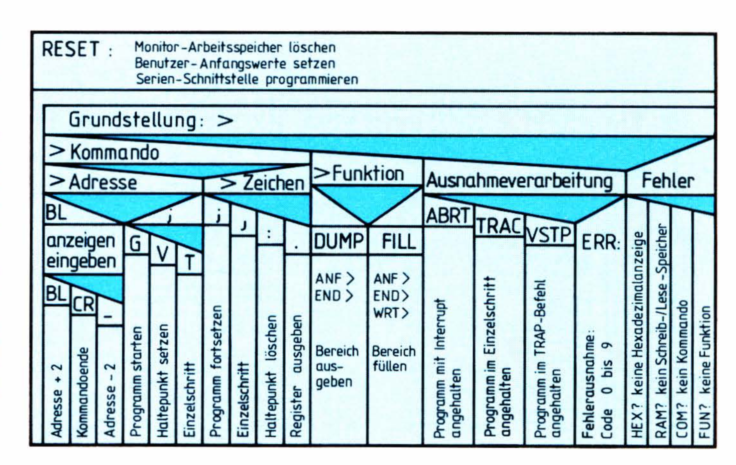


Bild 3. Struktogramm des Monitors

- Wurden vorher andere Zeichen eingegeben, so untersucht ein anderes Unterprogramm, ob die vier letzten Zeichen aus den Hexadezimalziffern 0...9 oder A...F bestanden. Ist dies der Fall, so handelte es sich um eine Adresse gefolgt von einem Kommandozeichen.
- Waren die vier letzten Zeichen keine Adresse, so wurde eine Funktion aufgerufen.

Die Verzweigung innerhalb dieser drei Fälle wird mit Sprungtabellen vorgenommen. Im Falle eines Funktionsaufrufs ist die Verzweigung einstufig; der Funktionsname steht zusammen mit der Sprungadresse in einer Tabelle. Bei Kommandos können mehrstufige Verzweigungen auftreten. Wird z. B. nach einer Adresse ein Semikolon eingegeben, so können in diesem Fall die drei Buchstaben G, V oder T folgen. Der Strichpunkt der ersten Stufe verzweigt in ein Programmstück, das ein weiteres Zeichen liest und damit in einer zweiten Stufe erneut verzweigt. In dieser Technik können beliebige Kombinationen von Kommandozeichen aufgebaut werden; lediglich das erste Kommandozeichen darf weder eine Ziffer noch ein Buchstabe sein. Gegenüber einer Verzweigungstechnik mit Vergleichsbefehlen sind Sprungtabellen wesentlich übersichtlicher und einfacher zu erweitern. Die Verzweigung der Ausnahmeverarbeitung übernimmt der Prozessor.

4 Programmdetails

Bild 4 zeigt das vollständige Programm als Übersetzungsliste eines Cross-Assemblers. Am linken Rand stehen Zeilennummern, auf die zur Erklärung der einzelnen Programmteile Bezug genommen wird.

Die Zeilen 40...420 enthalten die Ausnahmevektoren. Die Interruptvektoren 1...3 und 5...7 sowie die Befehle TRAP 0...TRAP E stehen dem Benutzer im RAM-Bereich zur Verfügung.

Die Zeilen 430...1160 bilden ein System von Unterprogrammen zur Ausgabe von Zeichen, Wörtern, Doppelwörtern und Registerinhalten.

```
                       * TERMINAL - MONITOR ELEKTRONIK 10.1.82 17:00
              00003401 ACIAST EQU    $3401    ACIA STEUERREGISTER
              00003403 ACIADA EQU    $3403    ACIA DATENREGISTER
                       * AUSNAHMEVEKTOREN
              00000000        ORG    $0000
       000000 000027A0 BEGIN  DC.L   SSTACK   MONITOR-STAPELZEIGER
       000004 000002A0        DC.L   START    MONITOR-STARTADRESSE
       000008 000004F8        DC.L   TER102   BUSFEHLER CODE 0
       00000C 00000518        DC.L   TER103   ADRESSFEHLER CODE 1
       000010 00000520        DC.L   TER104   OPERATIONSFEHLER CODE 2
       000014 00000528        DC.L   TER105   NULLDIVISION CODE 3
       000018 00000530        DC.L   TER106   CHK-BEFEHL CODE 4
       00001C 00000538        DC.L   TER107   TRAP-V-BEFEHL CODE 5
       000020 00000540        DC.L   TER108   STATUSVERLETZUNG CODE 6
       000024 000004D8        DC.L   TRAC     MONITOR-EINZELSCHRITT
       000028 00000548        DC.L   TER109   1010-EMULATOR CODE 7
       00002C 00000550        DC.L   TER110   1111-EMULATOR CODE 8
              00000060        ORG    $0060
       000060 00000558        DC.L   TER111   INTERRUPT-FEHLER CODE 9
       000064 000027EA        DC.L   UINT     BENUTZER-INTERRUPT 1
       000068 000027EA        DC.L   UINT     BENUTZER-INTERRUPT 2
       00006C 000027EA        DC.L   UINT     BENUTZER-INTERRUPT 3
       000070 0000048C        DC.L   ABRT     MONITOR-ABORT-TASTE
       000074 000027EA        DC.L   UINT     BENUTZER-INTERRUPT 5
       000078 000027EA        DC.L   UINT     BENUTZER-INTERRUPT 6
       00007C 000027EA        DC.L   UINT     BENUTZER-INTERRUPT 7
       000080 000027EE        DC.L   UTRA     BENUTZER-TRAP 0
       000084 000027EE        DC.L   UTRA     BENUTZER-TRAP 1
       000088 000027EE        DC.L   UTRA     BENUTZER-TRAP 2
       00008C 000027EE        DC.L   UTRA     BENUTZER-TRAP 3
       000090 000027EE        DC.L   UTRA     BENUTZER-TRAP 4
       000094 000027EE        DC.L   UTRA     BENUTZER-TRAP 5
       000098 000027EE        DC.L   UTRA     BENUTZER-TRAP 6
       00009C 000027EE        DC.L   UTRA     BENUTZER-TRAP 7
       0000A0 000027EE        DC.L   UTRA     BENUTZER-TRAP 8
       0000A4 000027EE        DC.L   UTRA     BENUTZER-TRAP 9
       0000A8 000027EE        DC.L   UTRA     BENUTZER-TRAP A
       0000AC 000027EE        DC.L   UTRA     BENUTZER-TRAP B
       0000B0 000027EE        DC.L   UTRA     BENUTZER-TRAP C
       0000B4 000027EE        DC.L   UTRA     BENUTZER-TRAP D
       0000B8 000027EE        DC.L   UTRA     BENUTZER-TRAP E
       0000BC 000004C2        DC.L   VSTP     MONITOR-HALTEPUNKT
                       * SYSTEM-UNTERPROGRAMME
              00000100        ORG    $0100
                       * AUS = AUSGEBEN 1 ZEICHEN AUS D0
       000100 1E383401 AUS    MOVE.B ACIAST,D7  STATUS LESEN
       000104 E41F            ROR.B  #2,D7      SENDERFLAG NACH C
       000106 64F8            BCC    AUS        SENDER VOLL
       000108 11C03403        MOVE.B D0,ACIADA  ZEICHEN NACH SENDER
       00010C 4E75            RTS
                       * AUSB4 = AUSGEBEN BL UND 4 ZEICHEN AUS D0
       00010E 2F00     AUSB4  MOVE.L D0,-(A7)   D0 RETTEN
       000110 103C0020        MOVE.B #$20,D0    BL LADEN
       000114 61EA            BSR    AUS        AUSGEBEN
       000116 201F            MOVE.L (A7)+,D0   D0 ZURUECK
                       * AUS4 = AUSGEBEN 4 ZEICHEN AUS D0
       000118 2F02     AUS4   MOVE.L D2,-(A7)   D2 RETTEN
       00011A 7404            MOVEQ  #4,D2      ZEICHENZAEHLER
       00011C E198     AUS5   ROL.L  #8,D0      ZEICHEN SCHIEBEN
       00011E 61E0            BSR    AUS        ZEICHEN AUSGEBEN
       000120 5342            SUBQ   #1,D2      ZAEHLEN
       000122 66F8            BNE    AUS5       WEITER
       000124 241F            MOVE.L (A7)+,D2   D2 ZURUECKLADEN
       000126 4E75            RTS
                       * AUSWD = AUSGEBEN BL UND WORT AUS D0
       000128 48E76000 AUSWD  MOVEM.L          D1-D2,-(A7) RETTEN
       00012C 7404            MOVEQ  #4,D2      ZAEHLER
       00012E 4840            SWAP   D0         WORT NACH VORN
       000130 2200     AUSWD1 MOVE.L D0,D1      WORT RETTEN NACH D1
       000132 103C0020        MOVE.B #$20,D0    BL NACH D0
       000136 61C8            BSR    AUS        BL AUSGEBEN
       000138 E999     AUSWD2 ROL.L  #4,D1      4 BIT SCHIEBEN
       00013A 1001            MOVE.B D1,D0      4 BIT NACH D0
       00013C 0200000F        ANDI.B #$0F,D0    MASKIEREN
       000140 06000030        ADDI.B #$30,D0    ADDIERE $30
       000144 0C00003A        CMPI.B #$3A,D0    A - F ?
       000148 65000006        BCS    AUSWD3     NEIN: ZIFFER
       00014C 06000007        ADDI.B #$07,D0    JA: ADDIERE $07
       000150 61AE     AUSWD3 BSR    AUS        ZEICHEN AUSGEBEN
       000152 5342            SUBQ   #1,D2      ZAEHLEN
       000154 66E2            BNE    AUSWD2     NEUER DURCHLAUF
       000156 4CDF0006        MOVEM.L          (A7)+,D1-D2 ZURUECKLADEN
       00015A 4E75            RTS
                       * AUSDW = AUSGEBEN BL UND DOPPELWORT AUS D0
       00015C 48E76000 AUSDW  MOVEM.L          D1-D2,-(A7) RETTEN
       000160 7408            MOVEQ  #8,D2      ZAEHLER
       000162 60CC            BRA    AUSWD1
                       * AUSRE = AUSGABE ALLER BENUTZER-REGISTER
       000164 41F827A0 AUSRE  LEA    UPC,A0     ANFANGSADRESSE REGISTER
       000168 203C2050433D    MOVE.L #$2050433D,D0   TEXT: PC=
       00016E 619E            BSR    AUSB4      AUSGEBEN
       000170 2018            MOVE.L (A0)+,D0   BEFEHLSZAEHLER
       000172 61E8            BSR    AUSDW      AUSGEBEN
       000174 203C2053523D    MOVE.L #$2053523D,D0   TEXT: SR=
       00017A 6192            BSR    AUSB4
       00017C 3018            MOVE.W (A0)+,D0   STATUSREGISTER
       00017E 61A8            BSR    AUSWD      AUSGEBEN
       000180 203C2053503D    MOVE.L #$2053503D,D0   TEXT: SP=
       000186 6186            BSR    AUSB4
       000188 2018            MOVE.L (A0)+,D0   S-STAPELZEIGER
       00018A 61D0            BSR    AUSDW      AUSGEBEN
       00018C 263C2044303D    MOVE.L #$2044303D,D3   TEXT: D0=
       000192 7802            MOVEQ  #2,D4      ZAEHLER D UND A REGISTER
       000194 7A02     AUSRE1 MOVEQ  #2,D5      2 ZEILEN
       000196 7C04     AUSRE2 MOVEQ  #4,D6      4 WERTE/ZEILE
       000198 203C0D0A2020    MOVE.L #$0D0A2020,D0   NEUE ZEILE
       00019E 6100FF6E        BSR    AUSB4      AUSGEBEN
       0001A2 2003     AUSRE3 MOVE.L D3,D0      REGISTERNUMMER
       0001A4 6100FF68        BSR    AUSB4      AUSGEBEN
       0001A8 06430100        ADD.W  #$0100,D3  NEUE REGISTERNUMMER
       0001AC 2018            MOVE.L (A0)+,D0   REGISTERINHALT
       0001AE 61AC            BSR    AUSDW      AUSGEBEN
       0001B0 5346            SUBQ   #1,D6      ZAEHLER WERTE/ZEILE
       0001B2 66EE            BNE    AUSRE3     WEITER
       0001B4 5345            SUBQ   #1,D5      ZAEHLER ZEILEN/REGISTERART
       0001B6 66DE            BNE    AUSRE2     WEITER
       0001B8 5344            SUBQ   #1,D4      REGISTERARTEN
       0001BA 6700000A        BEQ    AUSRE4     FERTIG
       0001BE 263C2041303D    MOVE.L #$2041303D,D3   TEXT: A0=
       0001C4 60CE            BRA    AUSRE1     WEITER
       0001C6 4E75     AUSRE4 RTS
                       * EINE = EINGABE EINES ZEICHENS NACH D0 OHNE ECHO
       0001C8 10383401 EINE   MOVE.B ACIAST,D0   STATUS LESEN
       0001CC E218            ROR.B  #1,D0      ZEICHEN DA?
       0001CE 64F8            BCC    EINE       NEIN
       0001D0 10383403        MOVE.B ACIADA,D0   ZEICHEN ABHOLEN
       0001D4 4E75            RTS
                       * EIN = EINGABE EINES ZEICHENS NACH D0 MIT ECHO
       0001D6 61F0     EIN    BSR    EINE
       0001D8 0200007F        ANDI.B #$7F,D0    MASKE BIT B7 = 0
       0001DC 0C000018        CMPI.B #$18,D0    CANCEL = CTR-X ?
       0001E0 6700010E        BEQ    TER2       ABBRUCH GRUNDSTELLUNG
       0001E4 0C000020        CMPI.B #$20,D0    STEUERZEICHEN?
       0001E8 6400FF16        BCC    AUS        NEIN: ECHO NACH TERMINAL
       0001EC 4E75            RTS               JA: KEIN ECHO
                       * EIN4 = EINGEBEN 4 ZEICHEN NACH D1
                       * ABBRUCHZEICHEN IN D0   ZAEHLER IN D2
       0001EE 223C30303030 EIN4 MOVE.L #´0000´,D1   MIT NULLEN VORBESETZT
       0001F4 4282            CLR.L  D2         ZAEHLER LOESCHEN
       0001F6 61DE     EIN41  BSR    EIN        ZEICHEN LESEN
       0001F8 0C000030        CMPI.B #´0´,D0  ZEICHEN 0 ?
       0001FC 65000022        BCS    EIN43      ABBRUCH BEI <0
       000200 0C00003A        CMPI.B #´:´,D0    > ZEICHEN 9 ?
       000204 65000012        BCS    EIN42      WEITER
       000208 0C000041        CMPI.B #´A´,D0    ZEICHEN A ?
       00020C 65000012        BCS    EIN43      ABBRUCH BEI < A
       000210 0C00005B        CMPI.B #$5B,D0    > ZEICHEN Z ?
       000214 6400000A        BCC    EIN43      ABBRUCH BEI > Z
       000218 E199     EIN42  ROL.L  #8,D1      EINGABE NACH LINKS
       00021A 1200            MOVE.B D0,D1      NEUES ZEICHEN
       00021C 5242            ADDQ   #1,D2      ZEICHENZAEHLER
       00021E 60D6            BRA    EIN41      NEUES ZEICHEN HOLEN
       000220 4E75     EIN43  RTS               ABBRUCH DER EINGABE
                       * DECO = UMWANDELN 4 ZEICHEN AUS D1 NACH D2 BINAER
                       * C = 0: HEXAZEICHEN  C = 1: KEINE HEXAZEICHEN
       000222 7604     DECO   MOVEQ  #4,D3    ZEICHENZAEHLER LADEN
       000224 2801            MOVE.L D1,D4      ZEICHEN RETTEN
       000226 E19C     DECO1  ROL.L  #8,D4      ZEICHEN SCHIEBEN
       000228 E94A            LSL.W  #4,D2      BINAERWERT SCHIEBEN
       00022A 0C040047        CMPI.B #´G´,D4    ZEICHEN G ?
       00022E 6400002C        BCC    DECO4      FEHLER
       000232 0C040041        CMPI.B #´A´,D4    ZEICHEN A ?
       000236 6500000E        BCS    DECO3      TESTEN ZIFFER
       00023A 04040037        SUBI.B #$37,D4    HEXAZIFFER A - F
       00023E 8404     DECO2  OR.B   D4,D2      ZUM BINAERWERT
       000240 5343            SUBQ   #1,D3      ZEICHENZAEHLER
       000242 66E2            BNE    DECO1      WEITER
       000244 4E75            RTS               FERTIG
       000246 0C04003A DECO3  CMPI.B #´:´,D4    > ZIFFER 9 ?
       00024A 64000010        BCC    DECO4      FEHLER
       00024E 0C040030        CMPI.B #´0´,D4    ZIFFER 0 ?
       000252 65000008        BCS    DECO4      FEHLER
       000256 04040030        SUBI.B #$30,D4    HEXAZIFFER 0 - 9
       00025A 60E2            BRA    DECO2      WEITER
       00025C 003C0001 DECO4  ORI.B  #1,CCR     C = 1: FEHLER
       000260 4E75            RTS
                       * EINAD = EINGEBEN ANFANGS- UND ENDADRESSE D5 UND D6
       000262 203C414E463E EINAD MOVE.L #´ANF>´,D0   TEXT: ANF>
       000268 6100FEA4        BSR    AUSB4      AUSGEBEN
       00026C 6100FF80        BSR    EIN4       ADRESSE LESEN
       000270 61B0            BSR    DECO       UMWANDELN
       000272 64000010        BCC    EINAD2     GUELTIG
       000276 203C4845583F EINAD1 MOVE.L #´HEX?´,D0   TEXT: HEX?
       00027C 6100FE90        BSR    AUSB4      FEHLERMELDUNG
       000280 6000006E        BRA    TER2       ABBRUCH GRUNDSTELLUNG
       000284 2A02     EINAD2 MOVE.L D2,D5      ANFANGSADRESSE
       000286 203C454E443E    MOVE.L #´END>´,D0   TEXT: END>
       00028C 6100FE80        BSR    AUSB4      AUSGEBEN
       000290 6100FF5C        BSR    EIN4       ADRESSE LESEN
       000294 618C            BSR    DECO       UMWANDELN
       000296 65EC            BCS    EINAD2     UNGUELTIG
       000298 2C02            MOVE.L D2,D6      ENDADRESSE
       00029A BC85            CMP.L  D5,D6      ENDE > ANFANG ?
       00029C 65C4            BCS    EINAD      FEHLER
       00029E 4E75            RTS
                       * HAUPTPROGRAMM
                       * ANFANGSWERTE IM RAM SETZEN
       0002A0 41F827A0 START  LEA    UPC,A0     MONITOR-RAM-ANFANG
       0002A4 4298     TER1   CLR.L  (A0)+      LOESCHEN RAM
       0002A6 B1FC00002800    CMPA.L #REND,A0   MONITOR-RAM-ENDE
       0002AC 66F6            BNE    TER1       WEITER
       0002AE 21FC00002000
                  27A0        MOVE.L #URAM,UPC  BENUTZER-RAM-ANFANG
       0002B6 31FC200027A4    MOVE.W #$2000,UCCR  BENUTZER-STATUSREGISTER
       0002BC 21FC00002780
                  27A6        MOVE.L #USTACK,USSP BENUTZER-S-STAPELZEIGER
       0002C4 21FC00002780
                  27E6        MOVE.L #USTACK,UREG+60  BENUTZER-U-STAPELZEIG
       0002CC 31FC4EF827EA    MOVE.W #$4EF8,UINT  JMP-BEFEHL
       0002D2 31FC4EF827EE    MOVE.W #$4EF8,UTRA  JMP-BEFEHL
       0002D8 31FC048C27EC    MOVE.W #ABRT,UINT+2 BENUTZER-INTERRUPT VORBES
       0002DE 31FC04C227F0    MOVE.W #VSTP,UTRA+2 BENUTZER-TRAP VORBESETZT
                       * SERIENSCHNITTSTELLE PROGRAMMIEREN
       0002E4 11FC00033401    MOVE.B #$03,ACIAST  RESET ACIA
       0002EA 11FC00153401    MOVE.B #$15,ACIAST  STEUERGROESSEN ACIA
                       * GRUNDSTELLUNG
       0002F0 4FF827A0 TER2   LEA    SSTACK,A7    MONITOR-STAPELZEIGER NEU
       0002F4 203C0D0A3E20    MOVE.L #$0D0A3E20,D0  NEUE ZEILE >
       0002FA 6100FE1C        BSR    AUS4           AUSGEBEN
       0002FE 6100FEEE        BSR    EIN4       BENUTZER-EINGABE
       000302 4A82            TST.L  D2         EINGABEZAEHLER ?
       000304 66000020        BNE    TER5       ADRESSE ODER FUNKTION PRUEF
                       * 1. ZEICHEN IST KOMMANDO IN D0
       000308 41F80798        LEA    TAB3,A0    TABELLE 3
                       * KOMMANDOTABELLE DURCHSUCHEN UND SPRINGEN
       00030C 3218     TER3   MOVE.W (A0)+,D1  ZEICHEN AUS TABELLE
       00030E 6600000E        BNE    TER4       VERGLEICHEN
       000312 203C434F4D3F    MOVE.L #´COM?´,D0  FEHLERMELDUNG: COM?
       000318 6100FDF4        BSR    AUSB4       AUSGEBEN
       00031C 60D2            BRA    TER2       ABBRUCH GRUNDSTELLUNG
       00031E 3258     TER4   MOVE.W (A0)+,A1   ADRESSE LADEN
       000320 B200            CMP.B  D0,D1      ZEICHEN VERGLEICHEN
       000322 66E8            BNE    TER3       WEITER SUCHEN
       000324 4ED1            JMP    (A1)       SPRINGEN
                       * ADRESSE ODER FUNKTION PRUEFEN
       000326 6100FEFA TER5   BSR    DECO       UMWANDELN
       00032A 65000016        BCS    TER7       WAR FUNKTION
                       * ADRESSE EINGEGEBEN
       00032E 08820000        BCLR   #0,D2      ADRESSE GERADE MACHEN
       000332 41F80780        LEA    TAB1,A0    TABELLE 1
       000336 60D4            BRA    TER3       DURCHSUCHEN UND SPRINGEN
                       * ADRESSE ;  EINGEGEBEN
       000338 6100FE9C TER6   BSR    EIN        2.ZEICHEN LESEN
       00033C 41F8078A        LEA    TAB2,A0    TABELLE 2
```

Bild 4. Das Monitorprogramm, erstellt mit einem Cross-Assembler

```
000340 60CA                 BRA     TER3         DURCHSUCHEN UND SPRINGEN
                    * FUNKTION EINGEGEBEN
000342 41F807AA      TER7   LEA     FTAB,A0      FUNKTIONSTABELLE
000346 2018          TER8   MOVE.L (A0)+,D0      NAMEN LESEN
000348 6600000E             BNE     TER9         VERGLEICHEN
00034C 203C46554E3F         MOVE.L #´FUN?´,D0     FEHLERMELDUNG : FUN?
000352 6100FDBA             BSR     AUSB4        AUSGEBEN
000356 6098                 BRA     TER2         ABBRUCH GRUNDSTELLUNG
000358 3258          TER9   MOVE.W (A0)+,A1      SPRUNGADRESSE
00035A B280                 CMP.L   D0,D1        NAMEN VERGLEICHEN
00035C 66E8                 BNE     TER8         WEITER SUCHEN
00035E 4ED1                 JMP     (A1)         SPRINGEN
                    * WORT EIN/AUSGABE
000360 2042          TER10  MOVE.L D2,A0         LAUFENDE WORTADRESSE
000362 3010          TER11  MOVE.W (A0),D0       WORT LESEN
000364 6100FDC2             BSR     AUSWD        AUSGEBEN
000368 103C002D             MOVE.B #´-´,D0       ZEICHEN: -
00036C 6100FD92             BSR     AUS          AUSGEBEN
000370 6100FE7C             BSR     EIN4         NEUES WORT LESEN
000374 4A82                 TST.L   D2           EINGABE-ZEICHEN-ZAEHLER
000376 6700002E             BEQ     TER13        KOMMANDO EINGEGEBEN
00037A 6100FEA6             BSR     DECO         ZEICHEN UMWANDELN
00037E 64000010             BCC     TER12        GUELTIGE EINGABE
000382 203C4845583F         MOVE.L #´HEX?´,D0     FEHLERMELDUNG: HEX?
000388 6100FD84             BSR     AUSB4        AUSGEBEN
00038C 6000FF62             BRA     TER2         ABBRUCH GRUNDSTELLUNG
000390 3082          TER12  MOVE.W D2,(A0)       WORT ABSPEICHERN
000392 B450                 CMP.W   (A0),D2      VERGLEICHEN
000394 67000010             BEQ     TER13        WAR RAM-BEREICH
000398 203C52414D3F         MOVE.L #´RAM?´,D0     FEHLERMELDUNG: RAM?
00039E 6100FD6E             BSR     AUSB4        AUSGEBEN
0003A2 6000FF4C             BRA     TER2         ABBRUCH GRUNDSTELLUNG
0003A6 D1FC00000002 TER13   ADDA.L #2,A0         NEUE WORTADRESSE
0003AC 0C00000D             CMPI.B #$0D,D0       CR ?
0003B0 6700FF3E             BEQ     TER2         GRUNDSTELLUNG
0003B4 0C000020             CMPI.B #$20,D0       LEERZEICHEN BL ?
0003B8 6700001E             BEQ     TER15        NEUE ZEILE
0003BC 0C00002D             CMPI.B #´-´,D0       ZEICHEN - ?
0003C0 67000010             BEQ     TER14        VORHERGEHENDE ADRESSE
0003C4 203C434F4D00         MOVE.L #´COM´,D0      FEHLERMELDUNG: COM?
0003CA 6100FD42             BSR     AUSB4        AUSGEBEN
0003CE 6000FF20             BRA     TER2         ABBRUCH GRUNDSTELLUNG
0003D2 91FC00000004 TER14   SUBA.L #4,A0         WORTADRESSE - 2
0003D8 203C200A0D2A TER15   MOVE.L #$200A0D2A,D0   NEUE ZEILE
0003DE 6100FD38             BSR     AUS4         AUSGEBEN
0003E2 3008                 MOVE.W A0,D0         LAUFENDE WORTADRESSE
0003E4 6100FD42             BSR     AUSWD        AUSGEBEN
0003E8 6000FF78             BRA     TER11        WORT AUSGEBEN
                    * HALTEPUNKTVERWALTUNG
0003EC 31C227F2      TER20  MOVE.W D2,VPKT       HALTEPUNKT SETZEN
0003F0 6000FEFE             BRA     TER2         GRUNDSTELLUNG
0003F4 21FC00000000
              27F2   TER21  MOVE.L #0,VPKT       HALTEPUNKT LOESCHEN
0003FC 6000FEF2             BRA     TER2         GRUNDSTELLUNG
                    * REGISTERVERWALTUNG
000400 6100FD62      TER22  BSR     AUSRE        ALLE REGISTER AUSGEBEN
000404 6000FEEA             BRA     TER2         GRUNDSTELLUNG
                    * PROGRAMM STARTEN
000408 21C227A0      TER30  MOVE.L D2,UPC        ADRESSE;G
00040C 0278FEFF27A4 TER31   ANDI.W #$FEFF,UCCR    ABORT-TASTE FREI
000412 4280                 CLR.L   D0           D0 LOESCHEN
000414 303827F2             MOVE.W VPKT,D0       HALTEPUNKTADRESSE
000418 6700001E             BEQ     TER33        KEIN HALTEPUNKT
00041C B0B827A0             CMP.L   UPC,D0       START BEI HALTEPUNKT ?
000420 6600000C             BNE     TER32        NEIN: WEITER
000424 0038008027F6         ORI.B   #$80,MARK    JA: MARKE SETZEN
00042A 60000034             BRA     TER36        1 EINZELSCHRITT
00042E 2040          TER32  MOVE.L D0,A0         HALTEPUNKTADRESSE
000430 31D027F4             MOVE.W (A0),VPKT+2    CODE RETTEN
000434 30BC4E4F             MOVE.W #$4E4F,(A0)    TRAP-F = $4E4F
000438 207827E6      TER33  MOVE.L UREG+60,A0     U-STAPELZEIGER
00043C 4E60                 MOVE.L A0,USP        LADEN
00043E 2E7827A6             MOVE.L USSP,A7       BENUTZER S-STAPELZEIGER
000442 2F3827A0             MOVE.L UPC,-(A7)      BENUTZER-BEFEHLSZAEHLER
000446 3F3827A4             MOVE.W UCCR,-(A7)     BENUTZER-STATUSREGISTER
00044A 4DF827AA             LEA     UREG,A6
00044E 4CDE3FFF             MOVEM.L         (A6)+,D0-D7/A0-A5  REGISTER
000452 2C56                 MOVE.L (A6),A6
000454 4E73                 RTE                  PROGRAMMSTART
                    * EINZELSCHRITT
000456 21C227A0      TER34  MOVE.L D2,UPC         ADRESSE;T
00045A 0238007F27F6 TER35   ANDI.B #$7F,MARK      EINZELSCHRITTKOMMANDO: MARK
000460 0078800027A4 TER36   ORI.W   #$8000,UCCR    TRACE-FLAG ON
000466 60D0                 BRA     TER33         PROGRAMM STARTEN
                    * AUSNAHME - VERARBEITUNG
                    * SAVE = ALLE REGISTER RETTEN
000468 21CE27E2      SAVE   MOVE.L A6,UREG+56
00046C 4E6E                 MOVE.L USP,A6
00046E 21CE27E6             MOVE.L A6,UREG+60
000472 4DF827E2             LEA     UREG+56,A6
000476 48E6FFFC             MOVEM.L         D0-D7/A0-A5,-(A6)
00047A 2C7827A6             MOVE.L USSP,A6
00047E 31DE27A4             MOVE.W (A6)+,UCCR
000482 21DE27A0             MOVE.L (A6)+,UPC
000486 21CE27A6             MOVE.L A6,USSP
00048A 4E75                 RTS

                    * ABORT - INTERRUPT - EINSPRUNG
00048C 21CF27A6      ABRT   MOVE.L A7,USSP       STAPELZEIGER
000490 4FF827A0             LEA     SSTACK,A7    MONITORSTAPELZEIGER
000494 61D2                 BSR     SAVE         REGISTER RETTEN
000496 2C3C41425254         MOVE.L #´ABRT´,D6 MELDUNG
00049C 303827F2      ABRT1  MOVE.W VPKT,D0       HALTEPUNKT ?
0004A0 67000008             BEQ     ABRT2        NEIN
0004A4 3240                 MOVEA   D0,A1        JA
0004A6 32B827F4             MOVE.W VPKT+2,(A1) CODE ZURUECK
0004AA 203C200D0A2A ABRT2   MOVE.L #$200D0A2A,D0  NEUE ZEILE *
0004B0 6100FC5C             BSR     AUSB4
0004B4 2006                 MOVE.L D6,D0         MELDUNG
0004B6 6100FC56             BSR     AUSB4        AUSGEBEN
0004BA 6100FCA8             BSR     AUSRE        REGISTER AUSGEBEN
0004BE 6000FE30             BRA     TER2         GRUNDSTELLUNG
                    * TRAP - HALTEPUNKT - EINSPRUNG
0004C2 21CF27A6      VSTP   MOVE.L A7,USSP       STAPELZEIGER
0004C6 4FF827A0             LEA     SSTACK,A7    MONITORSTAPELZEIGER
0004CA 619C                 BSR     SAVE         REGISTER RETTEN
0004CC 55B827A0             SUBQ.L #2,UPC        BEFEHLSZAEHLER - 2
0004D0 2C3C56535450         MOVE.L #´VSTP´,D6 MELDUNG
0004D6 60C4                 BRA     ABRT1
                    * TRACE - EINZELSCHRITT - EINSPRUNG
0004D8 21CF27A6      TRAC   MOVE.L A7,USSP       STAPELZEIGER
0004DC 4FF827A0             LEA     SSTACK,A7    MONITORSTAPELZEIGER
0004E0 6186                 BSR     SAVE         REGISTER RETTEN
0004E2 02787FFF27A4         ANDI.W #$7FFF,UCCR TRACE FLAG AUS
0004E8 103827F6             MOVE.B MARK,D0       MARKE ?
0004EC 6B00FF1E             BMI     TER31        WEITER
0004F0 2C3C54524143         MOVE.L #´TRAC´,D6     MELDUNG
0004F6 60B2                 BRA     ABRT2
                    * FEHLER - EINSPRUNG
0004F8 11FC003027F6 TER102  MOVE.B #´0´,MARK CODE 0
0004FE 508F          FEHL0  ADDQ.L #8,A7         STAPELZEIGER + 8
000500 21CF27A6      FEHL   MOVE.L A7,USSP       STAPELZEIGER
000504 4FF827A0             LEA     SSTACK,A7    MONITORSTAPELZEIGER
000508 6100FF5E             BSR     SAVE         REGISTER RETTEN
00050C 2C3C45525258         MOVE.L #´ERRX´,D6 MELDUNG
000512 1C3827F6             MOVE.B MARK,D6       CODE DAZU
000516 6092                 BRA     ABRT2
000518 11FC003127F6 TER103  MOVE.B #´1´,MARK CODE 1
00051E 60DE                 BRA     FEHL0        MIT KORREKTUR SP
000520 11FC003227F6 TER104  MOVE.B #´2´,MARK     CODE 2
000526 60D8                 BRA     FEHL
000528 11FC003327F6 TER105  MOVE.B #´3´,MARK     CODE 3
00052E 60D0                 BRA     FEHL
000530 11FC003427F6 TER106  MOVE.B #´4´,MARK     CODE 4
000536 60C8                 BRA     FEHL
000538 11FC003527F6 TER107  MOVE.B #´5´,MARK     CODE 5
00053E 60C0                 BRA     FEHL
000540 11FC003627F6 TER108  MOVE.B #´6´,MARK     CODE 6
000546 60B8                 BRA     FEHL
000548 11FC003727F6 TER109  MOVE.B #´7´,MARK     CODE 7
00054E 60B0                 BRA     FEHL
000550 11FC003827F6 TER110  MOVE.B #´8´,MARK     CODE 8
000556 60A8                 BRA     FEHL
000558 11FC003927F6 TER111  MOVE.B #´9´,MARK     CODE 9
00055E 60A0                 BRA     FEHL
                    * FUNKTIONEN
                    * SPEICHERBEREICH AUSDRUCKEN
000560 6100FD00      DUMP   BSR     EINAD        ANFANGS- UND ENDADRESSE
000564 020500F0             ANDI.B #$F0,D5       MASKIEREN
000568 2045                 MOVE.L D5,A0         ANFANGSADRESSE
00056A 0006000F             ORI.B   #$0F,D6
00056E 2246                 MOVE.L D6,A1         ENDADRESSE
000570 7608          DUMP1  MOVEQ   #8,D3        8 WOERTER PRO ZEILE
000572 203C200A0D2A         MOVE.L #$200A0D2A,D0   NEUE ZEILE
000578 6100FB94             BSR     AUSB4        AUSGEBEN
00057C 2008                 MOVE.L A0,D0         ANFANGSADRESSE
00057E 6100FBDC             BSR     AUSDW        AUSGEBEN
000582 3018          DUMP2  MOVE.W (A0)+,D0      WORT LESEN
000584 6100FBA2             BSR     AUSWD        AUSGEBEN
000588 5343                 SUBQ    #1,D3        WORTZAEHLER
00058A 66F6                 BNE     DUMP2        WEITER
00058C B3C8                 CMPA.L A0,A1         ENDADRESSE ERREICHT ?
00058E 64E0                 BCC     DUMP1        WEITER
000590 6000FD5E             BRA     TER2         GRUNDSTELLUNG
                    * SPEICHERBEREICH FUELLEN
000594 6100FCCC      FILL   BSR     EINAD        ANFANGS- UND ENDADRESSE
000598 020500FE             ANDI.B #$FE,D5       WORTADRESSE GERADE
00059C 2045                 MOVE.L D5,A0      ANFANGSADRESSE
00059E 2246                 MOVE.L D6,A1      ENDADRESSE
0005A0 203C5752543E         MOVE.L #´WRT>´,D0     MELDUNG: WRT>
0005A6 6100FB66             BSR     AUSB4        AUSGEBEN
0005AA 6100FC42             BSR     EIN4         WORT LESEN
0005AE 6100FC72             BSR     DECO         UMWANDLN
0005B2 64000010             BCC     FILL1        GUELTIG
0005B6 203C4845583F         MOVE.L #´HEX?´,D0  FEHLERMELDUNG: HEX?
0005BC 6100FB50             BSR     AUSB4     AUSGEBEN
0005C0 6000FD2E             BRA     TER2         GRUNDSTELLUNG
0005C4 30C2          FILL1  MOVE.W D2,(A0)+      WORT SPEICHERN
0005C6 B3C8                 CMPA.L A0,A1         VERGLEICHE ENDADRESSE
0005C8 64FA                 BCC     FILL1        WEITER
0005CA 6000FD24             BRA     TER2         GRUNDSTELLUNG
                    * TABELLEN
       00000780             ORG     $0780
                    * ADRESSE BL  ADRESSE;
000780 00            TAB1   DC.B    0,$20
000782 0360                 DC.W    TER10        ADRESSE LEERZEICHEN
000784 00                   DC.B    0,´;´
000786 0338                 DC.W    TER6         ADRESSE;
000788 0000                 DC.W    0            TABELLENENDE
                    * ;G  ;V  ;T
00078A 00            TAB2   DC.B    0,´G´
00078C 0408                 DC.W    TER30        ADRESSE;G
00078E 00                   DC.B    0,´V´
000790 03EC                 DC.W    TER20        ADRESSE;V
000792 00                   DC.B    0,´T´
000794 0456                 DC.W    TER34        ADRESSE;T
000796 0000                 DC.W    0            TABELLENENDE
                    * /   :   ;   .
000798 00            TAB3   DC.B    0,´/´
00079A 045A                 DC.W    TER35        EINZELSCHRITT
00079C 00                   DC.B    0,´:´
00079E 03F4                 DC.W    TER21        HALTEPUNKT LOESCHEN
0007A0 00                   DC.B    0,´;´
0007A2 040C                 DC.W    TER31        PROGRAMM STARTEN
0007A4 00                   DC.B    0,´.´
0007A6 0400                 DC.W    TER22        REGISTERAUSGABE
0007A8 0000                 DC.W    0            TABELLENENDE
                    * FUNKTIONSTABELLE
0007AA 44            FTAB   DC.L    ´DUMP´
0007AE 0560                 DC.W    DUMP
0007B0 46                   DC.L    ´FILL´
0007B4 0594                 DC.W    FILL
0007B6 4C                   DC.L    ´LADE´
0007BA 063A                 DC.W    LADE
0007BC 00000000             DC.L    0            TABELLENENDE
                    * BENUTZER - RAM -BEREICH
       00002000      URAM   EQU     $2000        BENUTZER-RAM
       00002780      USTACK EQU     $2780        BENUTZER-STAPEL
                    * MONITOR - RAM - BEREICH
       000027A0      SSTACK EQU     $27A0        MONITOR-STAPEL
       000027A0             ORG     $27A0
0027A0 00000004      UPC    DS.L    1            BENUTZER-BEFEHLSZAEHLER
0027A4 00000002      UCCR   DS.W    1            BENUTZER-STATUSREGISTER
0027A6 00000004      USSP   DS.L    1            BENUTZER-STAPELZEIGER
0027AA 00000040      UREG   DS.L    16           BENUTZER-REGISTER
0027EA 00000002      UINT   DS.W    1            JMP-BEFEHL
0027EC 00000002             DS.W    1            BENUTZER-INTERRUPT-ADRESSE
0027EE 00000002      UTRA   DS.W    1            JMP-BEFEHL
0027F0 00000002             DS.W    1            BENUTZER-TRAP-ADRESSE
0027F2 00000004      VPKT   DS.L    1            HALTEPUNKT ADRESSE CODE
0027F6 00000001      MARK   DS.B    1            MARKE/FEHLERCODE
0027F7 00000009      RES    DS.B    9            RESERVE
       00002800      REND   EQU     $2800        RAM-ENDE
                            END
```

Die Zeilen 1170...1870 bestehen aus Unterprogrammen zur Eingabe und Umcodierung von Zeichen und Wörtern sowie Adressen.

In den Zeilen 2000...2160 des Hauptprogramms werden nach einem Reset Anfangswerte gesetzt, und die Serien-Schnittstelle wird programmiert. Die Übertragung geschieht mit 8 Bit und einem Stoppbit ohne Parität und ohne Interrupt bei einem Taktverhältnis von 1:16. Bei anderen Übertragungsbedingungen muß ein anderes Steuerwort gewählt werden.

Der Einsprungpunkt TER2 der Zeile 2180 führt in die Grundstellung mit der Ausgabe eines „>“ am linken Rand. Da der Monitor-Stapelzeiger neu geladen wird, kann auch aus jedem beliebig tief geschachtelten Unterprogramm an diese Stelle gesprungen werden.

Die Zeilen 2210...2560 bilden den bereits erwähnten Sprungverteiler für Kommandos und Funktionen.

In den Zeilen 2570...3280 werden die einzelnen Kommandos ausgeführt. Ein Kommando kehrt entweder in die Grundstellung zum Einsprungpunkt TER2 zurück oder startet ein Benutzerprogramm und verläßt damit den Monitor.

Die Zeilen 3300...3890 behandeln die vom Monitor gesetzten Ausnahmen Interruptvektor-4 für die Abort-Taste, TRAP-F für einen Haltepunkt und TRACE für die Einzelschrittsteuerung. Ausnahmen, die auf Fehlermeldungen führen, werden in den Zeilen 3900...4250 behandelt.

In den Zeilen 4260...4610 liegen die Programme für die Funktionen DUMP und FILL.

Die Zeilen 6000...6330 enthalten die Sprungtabellen für Kommandos und Funktionen. Die LADE-Funktion dient zum Laden eines mit dem Cross-Assembler erstellten Programms und wird wegen ihrer zu starken Abhängigkeit von der verwendeten Anlage nicht weiter behandelt.

Die Zeilen 7000...7190 beschreiben den RAM-Bereich. Die Zeilen 7130...7160 sind die Einsprungpunkte für benutzereigene Interrupt- und TRAP-Vektoren.

Literatur

[1] 16-Bit Mikroprozessor Benützer-Handbuch. Firmenschrift Motorola MC68000UM(AD).

[2] 68000 Cross Macro Assembler Reference Manual. Firmenschrift Motorola M68KXASM(D3).

Prof. Dipl.-Ing. Günter Schmitt

Tastatur und Anzeigeeinheit für 68000-Computer

Der ab Seite 65 vorgestellte Mikrocomputer enthält eine Parallel-Schnittstelle, an die eine Hexadezimal-Tastatur und eine Siebensegment-Anzeigeeinheit angeschlossen werden können. Mit einem entsprechenden Monitorprogramm ist es möglich, Speicherinhalte anzuzeigen und zu verändern sowie Programme zu starten und zu testen. Der Beitrag stellt ein solches Monitorprogramm und die erforderliche Hardware vor, die notwendig sind, um diese Funktionen auszuführen.

1 Die Hardware

Bild 1 gibt einen Überblick über die Schaltung des Mikrocomputers. Zwei EPROM-Bausteine 2716 auf den Byteadressen $0000...$0FFF bilden den Festwertspeicher für die Ausnahmevektoren, das Monitorprogramm und Tabellen. Ein großer Teil des Speichers ist frei für Programmerweiterungen.

Vier RAM-Bausteine 2114 auf den Byteadressen $2000...$27FF bilden den Arbeitsspeicher für den Monitor und den Benutzer, der dort seine Programme ablegen kann.

Mit der Reset-Taste wird der Monitor gestartet; mit der Abort-Taste kann ein Benutzerprogramm abgebrochen werden.

Bild 2 zeigt den Anschluß der Tastatur und der Siebensegment-Anzeigeeinheit an die Parallel-Schnittstelle 6820 [1]. Die Anzeigeelemente (gemeinsame Anode) werden einzeln von einem 1-aus-16-Decodierer angesteuert; die Ausgänge PB0...PB7 steuern die Segmente. Die Einheit wird im Multiplexverfahren betrieben. Das G-Segment des links stehenden Elementes erhält einen zusätzlichen Vorwiderstand, da es während des Laufes eines Benutzerprogramms dauernd eingeschaltet ist. Die Decodierausgänge bilden gleichzeitig die Spalten der Tastatur; die Tastenzeilen liegen an den Eingängen PA7 und PA6.

2 Monitorfunktionen

Bild 3 zeigt die Anordnung der Anzeigeelemente und Tasten. Nach einem Reset geht der Monitor in die Grundstellung und meldet sich mit: - 6 8 0 0 0 - -

Die links stehende Stelle zeigt den Status (Betriebsart) des Monitors an. Dann folgen vier Stellen für Adressen und durch einen Punkt getrennt vier Stellen für Speicherinhalte, die wortweise angezeigt werden. Die drei Felder lassen sich gut durch verschiedenfarbige Anzeigeelemente unterscheiden. Alle Anzeigen und Eingaben sind Hexadezimalzahlen.

Die Statusanzeige zeigt nicht nur dem Benutzer, in welcher Betriebsart er sich gerade befindet, sondern sie wird auch vom Monitor ausgewertet. Mit dem hexadezimalen Tastenfeld (0...F) gibt der Benutzer je nach Betriebsart Adressen oder Speicherinhalte ein.

2.1 Funktionstasten und ihre Bedeutung

A: Bringt den Monitor in den Adreßeingabebetrieb (mit der Statusanzeige A), in dem Adressen eingegeben werden.

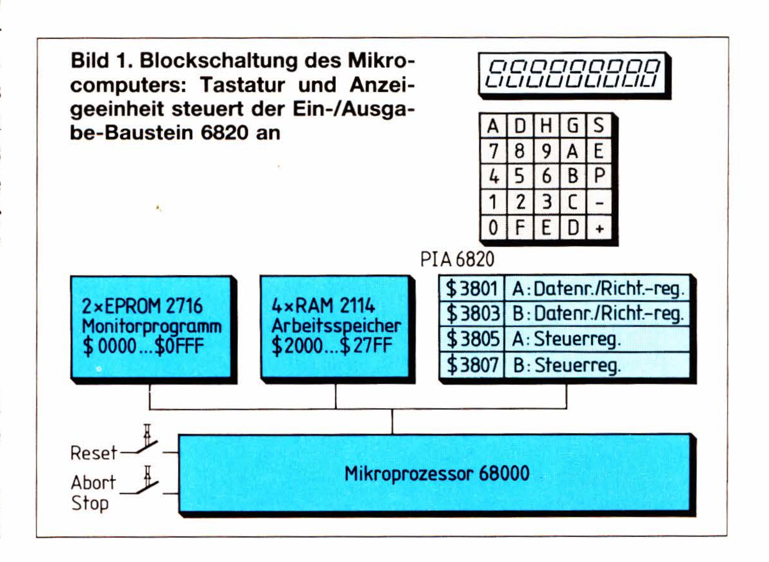


Bild 1. Blockschaltung des Mikrocomputers: Tastatur und Anzeigeeinheit steuert der Ein-/Ausgabe-Baustein 6820 an

D: Bringt den Monitor in den Dateneingabebetrieb (mit der Statusanzeige d), in dem der Inhalt des links adressierten Speicherwortes angezeigt und verändert werden kann.

H: Setzt die Statusanzeige auf H. Damit kann der Benutzer die Adresse eines Haltepunktes anzeigen und verändern.

P: Setzt die Statusanzeige auf P und zeigt damit den Befehlszähler des Benutzerprogramms an, der ebenfalls über die Hexadezimaltasten veränderbar ist.

+: Erhöht die angezeigte Adresse (Datenadresse, Haltepunktadresse oder Befehlszähler) um zwei.

–: Vermindert die Adresse um zwei.

G: Startet ein Programm beim augenblicklichen Stand des Befehlszählers.

E: Führt nur einen Befehl aus und dient damit der Verfolgung eines Programms im Einzelschritt.

S: Ist direkt an den IPL2-Eingang des Prozessors angeschlossen und löst einen Interrupt aus, mit dem ein Benutzerprogramm abgebrochen werden kann.

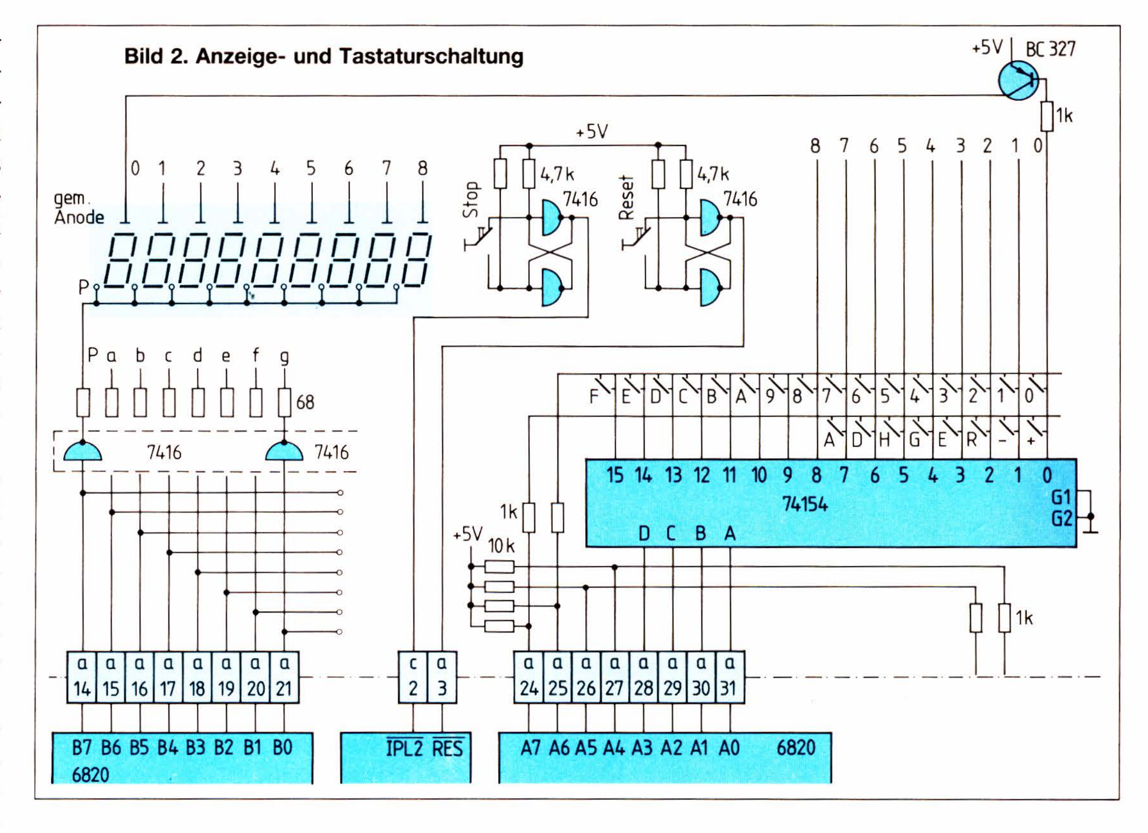
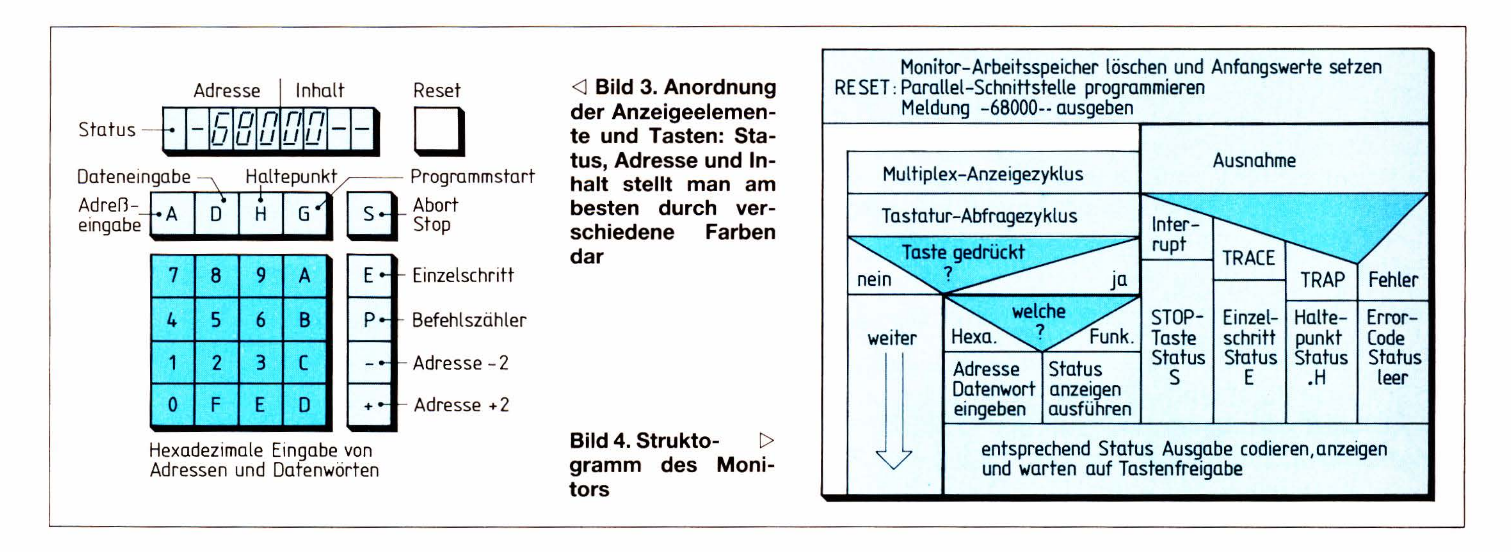


Bild 2. Anzeige- und Tastaturschaltung

2.2 Anzeigen und Ändern von Speicherwörtern

Mit der Funktionstaste A gelangt der Benutzer in den Adreßeingabebetrieb und gibt nun die Adresse des anzuzeigenden Speicherworts über die Hexadezimaltastatur ein. Die Stellen werden von rechts nach links durchgeschoben. Die Adresse wird beim Start des Monitors mit dem Wert 2000, dem Anfang des Arbeitsspeichers, vorbesetzt. Mit der Funktionstaste D wird in den Daten-Eingabebetrieb umgeschaltet. Im rechten Anzeigefeld erscheint der Inhalt des links adressierten Datenwortes. Der Benutzer kann nun:

– einen neuen Wert über die Tastatur eingeben und/oder
– die Adresse mit der „+"-Taste um zwei erhöhen und damit das nächste Wort bereit machen,
– die Adresse mit der „–"-Taste um zwei vermindern und damit das vorhergehende Wort bereit machen,
– mit einer anderen Funktionstaste eine neue Betriebsart wählen.



◁ Bild 3. Anordnung der Anzeigeelemente und Tasten: Status, Adresse und Inhalt stellt man am besten durch verschiedene Farben dar

Bild 4. Struktogramm des Monitors ▷

Damit ist es möglich, neue Programme und Daten einzugeben oder bestehende anzuzeigen.

2.3 *Starten eines Programms*

Mit der P-Taste wird der augenblickliche Stand des Benutzerbefehlszählers angezeigt. Diese Adresse kann mit der Hexadezimaltastatur neu eingegeben oder mit „+“ bzw. „–“ verändert werden. Die G-Taste startet mit diesem Wert das Benutzerprogramm. Dabei lädt der Monitor die Register aus dem Arbeitsbereich, setzt gegebenenfalls den Haltepunkt und gibt den Interrupt für die Abort-Taste frei. Das G-Segment der Statusanzeige wird als Marke eingeschaltet.

2.4 *Anhalten eines Programms*

Der Benutzer kann mit einem Sprungbefehl zur Adresse $0100 von seinem Programm in den Monitor zurückspringen. Auf der Ausgabe meldet sich der Monitor in der Grundstellung. Ein Benutzerprogramm kann durch die Abort-Taste mit einem Interrupt abgebrochen werden. Auf der Anzeige erscheinen der Status E, der augenblickliche Stand des Befehlszählers und der Code des Befehls, der als nächster ausgeführt wird.

2.5 *Setzen eines Haltepunktes*

Mit der H-Taste kann ein Haltepunkt an die Adresse eines Funktionscodes gesetzt werden; 0000 zeigt an, daß kein Haltepunkt gesetzt ist. Erreicht ein mit der G-Taste gestartetes Programm den Haltepunkt, so wird es mit dem Status .H abgebrochen. Angezeigt werden der augenblickliche Stand des Befehlszählers und der Code des Befehls, der als nächster ausgeführt wird.

2.6 *Einzelschritt*

Mit der E-Taste wird nur ein Befehl beim augenblicklichen Stand des Befehlszählers ausgeführt, und das Programm hält mit dem Status E und der Ausgabe des augenblicklichen Stands des Befehlszählers und des Codes des nächsten Befehls an.

2.7 *Registerverwaltung*

Der Monitor lädt bei jedem Programmstart die Register des Benutzers aus einem RAM-Bereich und schreibt sie bei jedem Programmabbruch mit Ausnahme des Reset dorthin wieder zurück. *Tabelle 1* zeigt die Adressen der Register, die vom Benutzer wie normale Speicherstellen gelesen und verändert werden können.

2.8 *Fehlermeldungen*

Bei der Ausführung eines Programms können Fehlermeldungen mit den in *Tabelle 2* zusammengestellten Codes auftreten.

Tabelle 1. Registeradressen im RAM-Bereich

Register	Adresse 1. Wort	Adresse 2. Wort	Bedeutung
D0	$27AA	$27AC	Datenregister
D1	$27AE	$27B0	Datenregister
D2	$27B2	$27B4	Datenregister
D3	$27B6	$27B8	Datenregister
D4	$27BA	$27BC	Datenregister
D5	$27BE	$27C0	Datenregister
D6	$27C2	$27C4	Datenregister
D7	$27C6	$27C8	Datenregister
A0	$27CA	$27CC	Adreßregister
A1	$27CE	$27D0	Adreßregister
A2	$27D2	$27D4	Adreßregister
A3	$27D6	$27D8	Adreßregister
A4	$27DA	$27DC	Adreßregister
A5	$27DE	$27E0	Adreßregister
A6	$27E2	$27E4	Adreßregister
A7	$27E6	$27E8	User-Stapelzeiger
SP	$27A6	$27A8	System-Stapelzeiger
SR	$27A4	—	Statusregister

3 Programmstruktur

Nach dem Setzen von Anfangswerten und nach der Ausgabe einer Meldung folgt eine aus drei Unterprogrammen bestehende Verarbeitungsschleife entsprechend dem Struktogramm (*Bild 4*). Auf einen Anzeigezyklus folgt ein Tastaturabfragezyklus. Wurde eine Taste betätigt, folgt die Auswertung entsprechend dem Struktogramm (*Bild 5*).

War eine Hexadezimaltaste gedrückt, so wird entsprechend dem Status ein Speicherwort, eine Speicheradresse, eine Haltepunktadresse oder der Befehlszähler verändert. Der neue Wert wird umcodiert und in einer Schleife angezeigt, bis die Taste freigegeben ist.

War eine Funktionstaste gedrückt, so wird die entsprechende Funktion ausgeführt, und es wird der entsprechende Status zusammen mit dem neuen Wert angezeigt. Auch hier wird auf die Freigabe der Taste gewar-

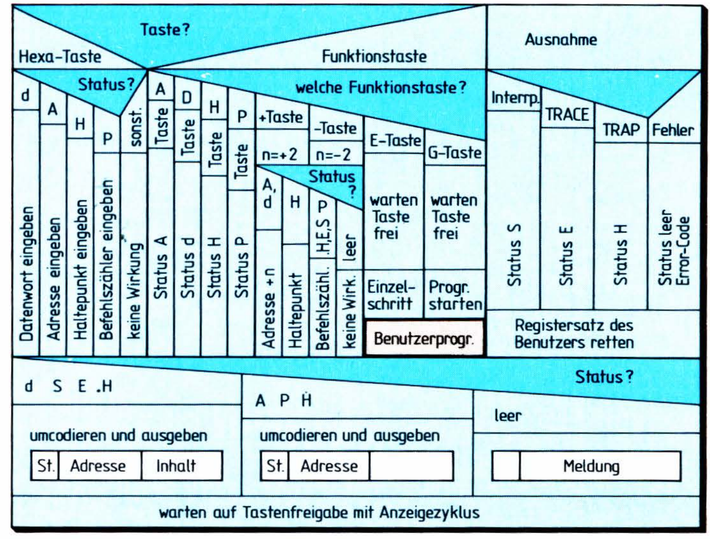


Bild 5. Auswertung der Tastendrücke und Ausgabe der Werte

Tabelle 2. Fehlermeldungen des Monitors

Meldung	Fehlerursache
Error 0	Busfehler
Error 1	Adreßfehler
Error 2	Operationsfehler
Error 3	Division durch Null
Error 4	CHK-Befehl
Error 5	TRAP-V-Befehl
Error 6	Statusverletzung
Error 7	1010-Emulator
Error 8	1111-Emulator
Error 9	Interrupt-Fehler

tet. Die G- bzw. E-Funktionen warten zuerst die Tastenfreigabe ab und starten dann das Programm. Bei einer Ausnahmeverarbeitung wird der entsprechende Status mit dem Befehlszählerstand und dem Code angezeigt, bzw. es wird im Fehlerfall ein Fehlercode ausgegeben.

4 Programmdetails

Bild 6 zeigt den Ausdruck des Monitors, erstellt von einem Cross-Assembler. Auf die am linken Rand stehenden Zeilennummern wird bei der Erklärung der Programmteile Bezug genommen.

Die Zeilen 80...440 enthalten die Ausnahmevektoren, von denen der Monitor die Einzelschritt-Ausnahme, die Auto-Interrupt-4-Ausnahme und die TRAP-F-Ausnahme für die Testhilfen benutzt.

Das Hauptprogramm der Zeilen 450...770 setzt zunächst Anfangswerte und arbeitet dann in einer Schleife mit dem Aufruf von drei Unterprogrammen.

```
  10                   * TASTEN - MONITOR ELEKTRONIK 11.1.82  14:00
  20         00003801  PIAAD  EQU    $3801    PIA-A-DATEN/RICHTUNG
  30         00003805  PIAAC  EQU    $3805    PIA-A-CONTROLLREG.
  40         00003803  PIABD  EQU    $3803    PIA-B-DATEN/RICHTUNG
  50         00003807  PIABC  EQU    $3807    PIA-B-CONTROLLREG.
  60                   * AUSNAHMEVEKTOREN
  70         00000000         ORG    $0000    E P R O M
  80 000000 0000279E   BEGIN  DC.L   SSTACK   MONITOR-STAPELZEIGER
  90 000004 00000100          DC.L   START    MONITOR-STARTADRESSE
 100 000008 000003AE          DC.L   TAS102   BUSFEHLER CODE 0
 110 00000C 000003BA          DC.L   TAS103   ADRESSFEHLER CODE 1
 120 000010 000003C6          DC.L   TAS104   OPERATIONSFEHLER CODE 2
 130 000014 000003D0          DC.L   TAS105   NULLDIVISION CODE 3
 140 000018 000003DA          DC.L   TAS106   CHK-BEFEHL CODE 4
 150 00001C 000003E4          DC.L   TAS107   TRAP-V-BEFEHL CODE 5
 160 000020 000003EE          DC.L   TAS108   STATUSVERLETZUNG CODE 6
 170 000024 0000038E          DC.L   TRAC     MONITOR - EINZELSCHRITT
 180 000028 000003F8          DC.L   TAS109   1010-EMULATOR CODE 7
 190 00002C 00000402          DC.L   TAS110   1111-EMULATOR CODE 8
 200         00000060         ORG    $0060    E P R O M
 210 000060 0000040C          DC.L   TAS111   INTERRUPT-FEHLER CODE 9
 220 000064 000027EE          DC.L   UINT     BENUTZER-INTERRUPT 1
 230 000068 000027EE          DC.L   UINT     BENUTZER-INTERRUPT 2
 240 00006C 000027EE          DC.L   UINT     BENUTZER-INTERRUPT 3
 250 000070 0000034C          DC.L   ABRT     MONITOR - ABORT -TASTE
 260 000074 000027EE          DC.L   UINT     BENUTZER-INTERRUPT 5
 270 000078 000027EE          DC.L   UINT     BENUTZER-INTERRUPT 6
 280 00007C 000027EE          DC.L   UINT     BENUTZER-INTERRUPT 7
 290 000080 000027F2          DC.L   UTRA     BENUTZER-TRAP 0
 300 000084 000027F2          DC.L   UTRA     BENUTZER-TRAP 1
 310 000088 000027F2          DC.L   UTRA     BENUTZER-TRAP 2
 320 00008C 000027F2          DC.L   UTRA     BENUTZER-TRAP 3
 330 000090 000027F2          DC.L   UTRA     BENUTZER-TRAP 4
 340 000094 000027F2          DC.L   UTRA     BENUTZER-TRAP 5
 350 000098 000027F2          DC.L   UTRA     BENUTZER-TRAP 6
 360 00009C 000027F2          DC.L   UTRA     BENUTZER-TRAP 7
 370 0000A0 000027F2          DC.L   UTRA     BENUTZER-TRAP 8
 380 0000A4 000027F2          DC.L   UTRA     BENUTZER-TRAP 9
 390 0000A8 000027F2          DC.L   UTRA     BENUTZER-TRAP A
 400 0000AC 000027F2          DC.L   UTRA     BENUTZER-TRAP B
 410 0000B0 000027F2          DC.L   UTRA     BENUTZER-TRAP C
 420 0000B4 000027F2          DC.L   UTRA     BENUTZER-TRAP D
 430 0000B8 000027F2          DC.L   UTRA     BENUTZER-TRAP E
 440 0000BC 00000378          DC.L   VSTP     MONITOR - HALTEPUNKT
 450                   * HAUPTPROGRAMM
 460         00000100         ORG    $0100    E P R O M
 470                   * ANFANGSWERTE IM RAM SETZEN
 480 000100 41F827A0   START  LEA    UPC,AO   MONITOR-RAM-ANFANG
 490 000104 4258       TAS1   CLR    (AO)+    RAM LOESCHEN
 500 000106 B1FC00002800      CMPA.L #REND,AO MONITOR-RAM-ENDE
 510 00010C 66F6              BNE    TAS1     .WEITER
 515 00010E 31FC2000279E      MOVE.W #URAM,LADD LAUFENDE ADRESSE
 520 000114 21FC00002000
                27A0          MOVE.L #URAM,UPC BENUTZER-RAM-ANFANG
 530 00011C 31FC200027A4      MOVE.W #$2000,UCCR BENUTZER-STATUSREGISTER
 540 000122 21FC00002780
                27A6          MOVE.L #USTACK,USSP BENUTZER-S-STAPELZEIGER
 550 00012A 21FC00002780
                27E6          MOVE.L #USTACK,UREG+60 BENUTZER-U-STAPELZEIG
 560 000132 31FC4EF827EE      MOVE.W #$4EF8,UINT JMP-BEFEHL
 570 000138 31FC4EF827F2      MOVE.W #$4EF8,UTRA JMP-BEFEHL
 580 00013E 31FC034C27F0      MOVE.W #ABRT,UINT+2 BENUTZER-INTERRUPT
 590 000144 31FC037827F4      MOVE.W #VSTP,UTRA+2 BENUTZER-TRAP
 600                   * PARALLELSCHNITTSTELLE PROGRAMMIEREN
 610 00014A 11FC000F3801      MOVE.B #$0F,PIAAD  RICHTUNG A SEITE
 615 000150 11FC00FF3803      MOVE.B #$FF,PIABD  RICHTUNG B SEITE
 620 000156 11FC00043805      MOVE.B #$04,PIAAC  DATENREGISTER EIN
 625 00015C 11FC00043807      MOVE.B #$04,PIABC  DATENREGISTER EIN
 640                   * GRUNDSTELLUNG
 650 000162 423827F7   TAS2   CLR.B  AUS      STATUS LEER
 660 000166 21FC015F7F7E
                27F8          MOVE.L #$015F7F7E,AUS+1  -680
 670 00016E 21FC7E7E0101
                27FC          MOVE.L #$7E7E0101,AUS+5  00--
 720                   * VERARBEITUNGSSCHLEIFE
 730 000176 6100000E   TAS4   BSR    ANZ      ANZEIGE
 740 00017A 6100002E          BSR    TEST     TASTEN PRUEFEN
 750 00017E 64F6              BCC    TAS4     KEINE TASTE GEDRUECKT
 760 000180 61000080          BSR    TASTE    TASTE GEDRÜCKT: AUSFUEHREN
 770 000184 60F0              BRA    TAS4     SCHLEIFE
 780                   * ANZ = ANZEIGE - UNTERPROGRAMM
 790 000186 4DF827F7   ANZ    LEA    AUS,A6   ADRESSE AUSGABEBEREICH
 800 00018A 7E09              MOVEQ  #9,D7    DURCHLAUFZAEHLER
 810 00018C 42383801          CLR.B  PIAAD    LINKE STELLE BEGINNT
 820 000190 11DE3803   ANZ1   MOVE.B (A6)+,PIABD  CODE AUSGEBEN
 830 000194 3C3C0200          MOVE.W #$0200,D6  WARTEZAEHLER
 840 000198 5346       ANZ2   SUBQ   #1,D6      ZAEHLER - 1
 850 00019A 66FC              BNE    ANZ2       BIS NULL
 860 00019C 42383803          CLR.B  PIABD      ANZEIGE DUNKEL
 870 0001A0 52383801          ADDQ.B #1,PIAAD   EINE STELLE WEITER
 880 0001A4 5347              SUBQ   #1,D7      DURCHLAUFZAEHLER -1
 890 0001A6 66E8              BNE    ANZ1       NAECHSTE STELLE
 900 0001A8 4E75              RTS               FERTIG
 910                   * TEST = TASTEN PRUEFEN
 920                   * C=0: KEINE  C=1: JA  DO=CODE  D1= ZEILE
 930 0001AA 4280       TEST   CLR.L  DO       ANFANGSWERT
 940 0001AC 11C03801   TEST1  MOVE.B DO,PIAAD SPALTE AUSGEBEN
 950 0001B0 12383801          MOVE.B PIAAD,D1 ZEILE LESEN
 960 0001B4 020100C0          ANDI.B #$C0,D1  MASKIEREN
 970 0001B8 0C0100C0          CMPI.B #$C0,D1  VERGLEICHEN
 980 0001BC 6600000C          BNE    TEST2    GEDRUECKT
 990 0001C0 5200              ADDQ.B #1,DO    NEUE SPALTE
1000 0001C2 0C000010          CMPI.B #$10,DO  ENDE ?
1010 0001C6 66E4              BNE    TEST1    NEIN: WEITER
1020 0001C8 4E75              RTS             JA: C=0 KEINE TASTE
1030 0001CA 003C0001   TEST2  ORI.B  #1,CCR   C=1: TASTE GEDRUECKT
1040 0001CE 4E75              RTS
1050                   * UMCO = WORT AUS DO NACH 4 ZEICHEN IN D1 UMCODIEREN
1060 0001D0 4DF8043E   UMCO   LEA    COTAB,A6 ADRESSE CODETABELLE
1070 0001D4 7E04              MOVEQ  #4,D7    ZAEHLER 4 ZEICHEN
1080 0001D6 E958       UMCO1  ROL.W  #4,DO    4 BIT SCHIEBEN
1090 0001D8 E189              LSL.L  #8,D1    ZEICHEN SCHIEBEN
1100 0001DA 3C00              MOVE.W DO,D6    WORT NACH D6
1110 0001DC 0246000F          ANDI.W #$000F,D6  4 BIT MASKIEREN
1120 0001E0 12366000          MOVE.B 0(A6,D6),D1  TABELLENZUGRIFF
1130 0001E4 5347              SUBQ   #1,D7    ZAEHLER - 1
1140 0001E6 66EE              BNE    UMCO1    BIS ZAEHLER NULL
1150 0001E8 4E75              RTS             FERTIG
1160                   * AUSWD = DATENWORT RECHTS ANZEIGEN
1170 0001EA 3011       AUSWD  MOVE.W (A1),DO  WORT NACH DO
1180 0001EC 61E2              BSR    UMCO     UMCODIEREN
1190 0001EE 21C127FC          MOVE.L D1,AUS+5 DATENWORT ANZEIGEN
1210                   * AUSAD = ADRESSE LINKS ANZEIGEN
1220 0001F2 3010       AUSAD  MOVE.W (AO),DO  ADRESSE NACH DO
1230 0001F4 61DA              BSR    UMCO     UMCODIEREN
1240 0001F6 21C127F8          MOVE.L D1,AUS+1 ADRESSE ANZEIGEN
1260                   * WAIT = WARTEN MIT ANZEIGE BIS TASTE FREI
1270 0001FA 618A       WAIT   BSR    ANZ      ANZEIGEZYKLUS
1280 0001FC 61AC              BSR    TEST     TASTEN PRUEFEN
1290 0001FE 65FA              BCS    WAIT     TASTE GEDRUECKT
1300 000200 4E75              RTS             TASTE FREI
1310                   * TASTE = TASTE AUSWERTEN
1320 000202 4A01       TASTE  TST.B  D1       ZEILE PRUEFEN
1330 000204 6A000044          BPL    FUNK     A7=0: FUNKTIONSTASTE
1340                   * HEXADEZIMALTASTE: STATUS PRUEFEN
1350 000208 0C38003D27F7 HEX1 CMPI.B #$3D,AUS D - STATUS ?
1360 00020E 66000010          BNE    HEX2
1370 000212 3278279E          MOVEA.W         LADD,A1  LAUFENDE ADRESSE
1380 000216 3211              MOVE.W (A1),D1  ALTES WORT LADEN
1390 000218 E949              LSL.W  #4,D1    4 BIT LINKS
1400 00021A 8200              OR.B   DO,D1    NEUE STELLE RECHTS
1410 00021C 3281              MOVE.W D1,(A1)  NEUES WORT SPEICHERN
1420 00021E 60CA              BRA    AUSWD    WORT ANZEIGEN UND WARTEN
1440 000220 0C38007727F7 HEX2 CMPI.B #$77,AUS A - STATUS ?
1450 000226 67000018          BEQ    HEX3
1460 00022A 0C38003727F7      CMPI.B #$37,AUS H - STATUS ?
1470 000230 6700000E          BEQ    HEX3
1480 000234 0C38006727F7      CMPI.B #$67,AUS P - STATUS ?
1490 00023A 67000004          BEQ    HEX3
1500 00023E 4E75              RTS             KEINE WIRKUNG
1510 000240 3210       HEX3   MOVE.W (AO),D1  ALTE ADRESSE LADEN
1520 000242 E949              LSL.W  #4,D1    4 BIT LINKS
1530 000244 8200              OR.B   DO,D1    NEUE STELLE RECHTS
1540 000246 3081              MOVE.W D1,(AO)  NEUE ADRESSE SPEICHERN
1550 000248 60A8              BRA    AUSAD    ADRESSEN ANZEIGEN UND WARTEN
1570                   * FUNKTIONSTASTE SPRUNGVERTEILER
1580 00024A 43F8044E   FUNK   LEA    FUTAB,A1 ADRESSE SPRUNGTABELLE
1590 00024E E308              LSL.B  #1,DO    CODE X 2
1600 000250 32710000          MOVE.W 0(A1,DO),A1 SPRUNGADRESSE NACH A1
1610 000254 4ED1              JMP    (A1)     SPRUNG
1620                   * A - TASTE = ADRESSEINGABE
1630 000256 11FC007727F7 FUNAD MOVE.B #$77,AUS STATUS  A
1640 00025C 41F8279E          LEA    LADD,AO  LAUFENDE ADRESSE
1650 000260 42B827FC          CLR.L  AUS+5    DATENFELD LEER
1660 000264 608C              BRA    AUSAD    ADRESSE ANZEIGEN UND WARTEN
1680                   * D - TASTE = DATENEINGABE
1690 000266 11FC003D27F7 FUNDA MOVE.B #$3D,AUS STATUS  D
1700 00026C 41F8279E          LEA    LADD,AO  LAUFENDE ADRESSE
1720 000270 3250              MOVE.W (AO),A1  DATENWORT LADEN
1730 000272 6000FF76          BRA    AUSWD    DATENWORT ANZEIGEN UND WARTEN
1750                   * H - TASTE = HALTEPUNKTEINGABE
1760 000276 11FC003727F7 FUNHP MOVE.B #$37,AUS STATUS  H
1770 00027C 41F827EA          LEA    VPKT,AO  HALTEPUNKTADRESSE
1780 000280 42B827FC          CLR.L  AUS+5    DATENFELD LEER
```

Das Unterprogramm ANZ der Zeilen 780...900 zeigt in einem Durchlauf alle neun Siebensegmentstellen an.

Das Unterprogramm TEST der Zeilen 910...1040 fragt die Tastatur ab. Durch das vorhergehende Anzeigeprogramm wird die Tastatur entprellt.

Die Unterprogramme UMCO und die drei folgenden Unterprogramme der Zeilen 1050...1300 werden für die Tastenauswertung benötigt.

Das Unterprogramm TASTE der Zeilen 1310...2250 unterscheidet zwischen den Hexadezimaltasten am Eingang PA6 und den Funktionstasten am Eingang PA7. Nach der Auswertung wird auf die Freigabe der Taste im Unterprogramm WAIT gewartet.

In den Zeilen 2260...2690 verarbeitet der Monitor die Ausnahmen der Abort-Taste, des Haltepunktes und des Einzelschritts. Die Zeilen 2700...3040 behandeln die Fehler-Ausnahmen.

Die Zeilen 7000...7270 enthalten die Codetabelle für den Siebensegmentcode und die Sprungtabelle für die Funktionstasten.

Die Zeilen 8000...8110 beschreiben den Arbeitsspeicher des Monitors. In den Zeilen 8110...8140 liegen die Einsprungpunkte für benutzereigene Interrupt- und Trap-Vektoren.

Literatur

[1] *Schmitt, G.*: Grundlagen der Mikrocomputertechnik. Oldenbourg Verlag, München 1981.

[2] 16-Bit-Mikroprozessor-Benützer-Handbuch. Firmenschrift, Motorola MC68000UM(AD).

[3] 68000 Cross Macro Assembler Reference Manual. Firmenschrift, Motorola M68KXASM(D3).


```
1790 000284 6000FF6C               BRA     AUSAD     ADRESSE ANZEIGEN UND WARTEN
1810                        * P - TASTE = BEFEHLSZAEHLEREINGABE
1820 000288 11FC006727F7 FUNPC  MOVE.B #$67,AUS STATUS  P
1830 00028E 41F827A2            LEA    UPC+2,A0 BEFEHLSZAEHLER
1840 000292 42B827FC            CLR.L  AUS+5    DATENFELD LEER
1850 000296 6000FF5A            BRA    AUSAD    ADRESSE ANZEIGEN UND WARTEN
1870                        * PLUS - TASTE = ADRESSE + 2
1880 00029A 7602         FUNPL  MOVEQ  #2,D3    KONSTANTE +2
1890 00029C 60000004            BRA    FUNPM
1900                        * MINUS - TASTE = ADRESSE - 2
1910 0002A0 76FE         FUNMI  MOVEQ  #-2,D3   KONSTANTE - 2
1920 0002A2 4A3827F7     FUNPM  TST.B  AUS      STATUS ?
1921 0002A6 66000004            BNE    FUNP1    NEIN: WEITER
1922 0002AA 4E75                RTS             LEER: KEINE WIRKUNG
1923 0002AC D750         FUNP1  ADD.W  D3,(A0)  ADRESSE ERHOEHEN
1925 0002AE 4A3827FC            TST.B  AUS+5    STATUS A,H,P ?
1926 0002B2 6700FF3E            BEQ    AUSAD    ADRESSEN ANZEIGEN UND WARTEN
1927 0002B6 3250                MOVE.W (A0),A1
1928 0002B8 6000FF30            BRA    AUSWD    WORT ANZEIGEN UND WARTEN
1960                        * G - TASTE = PROGRAMMSTART
1970 0002BC 0278FEFF27A4 FUNGO  ANDI.W #$FEFF,UCCR ABORT-TASTE FREI
1980 0002C2 4280                CLR.L  D0       LOESCHEN
1990 0002C4 303827EA            MOVE.W VPKT,D0  HALTEPUNKTADRESSE
2000 0002C8 6700001E            BEQ    FUN9     KEIN HALTEPUNKT
2010 0002CC B0B827A0            CMP.L  UPC,D0   START BEI HALTEPUNKT ?
2020 0002D0 6600000C            BNE    FUN8     NEIN: WEITER
2030 0002D4 0038008027F6        ORI.B  #$80,MARK JA: MARKE SETZEN
2040 0002DA 60000044            BRA    FUN12    1 EINZELSCHRITT
2050 0002DE 2040         FUN8   MOVE.L D0,A0    HALTEPUNKTADRESSE
2060 0002E0 31D027EC            MOVE.W (A0),VPKT+2 CODE RETTEN
2070 0002E4 30BC4E4F            MOVE.W #$4E4F,(A0) TRAP-F-CODE
2080 0002E8 6100FF10     FUN9   BSR    WAIT     WARTEN BIS TASTE FREI
2085 0002EC 207827E6            MOVE.L UREG+60,A0 U-STAPELZEIGER
2090 0002F0 4E60                MOVE.L A0,USP   LADEN
2100 0002F2 2E7827A6            MOVE.L USSP,A7  BENUTZER-S-STAPELZEIGER
2105 0002F6 023800FE27A3        ANDI.B #$FE,UPC+3 ADRESSE GERADE
2110 0002FC 2F3827A0            MOVE.L UPC,-(A7) BENUTZER-BEFEHLSZAEHLER
2120 000300 3F3827A4            MOVE.W UCCR,-(A7) BENUTZER-STATUSREGISTER
2130 000304 4DF827AA            LEA    UREG,A6  REGISTERADRESSEN
2140 000308 4CDE3FFF            MOVEM.L         (A6)+,D0-D7/A0-A5 REGISTER
2150 00030C 2C56                MOVE.L (A6),A6  REGISTER A6 LADEN
2170 00030E 42383801            CLR.B  PIAAD    LINKE STELLE
2180 000312 11FC00013803        MOVE.B #$01,PIABD  - ANZEIGEN
2190 000318 4E73                RTE             START
2220                        * E - TASTE = EINZELSCHRITT
2240 00031A 0238007F27F6 FUNES  ANDI.B #$7F,MARK MARKE SETZEN
2245 000320 0078800027A4 FUN12  ORI.W  #$8000,UCCR TRACE FLAG AN
2250 000326 60C0                BRA    FUN9     STARTEN
2260                        * AUSNAHME - VERARBEITUNG
2270                        * SAVE = REGISTER RETTEN
2280 000328 21CE27E2     SAVE   MOVE.L A6,UREG+56   A6 REGISTER
2290 00032C 4E6E                MOVE.L USP,A6       BENUTZER U-STAPELZEIGER
2300 00032E 21CE27E6            MOVE.L A6,UREG+60   BENUTZER U-STAPELZEIGER
2310 000332 4DF827E2            LEA    UREG+56,A6   ADRESSE REGISTERBEREICH
2320 000336 48E6FFFC            MOVEM.L         D0-D7/A0-A5,-(A6) REGISTER
2330 00033A 2C7827A6            MOVE.L USSP,A6      S-STAPELZEIGER
2340 00033E 31DE27A4            MOVE.W (A6)+,UCCR     BEDINGUNGSREGISTER
2350 000342 21DE27A0            MOVE.L (A6)+,UPC      BEFEHLSZAEHLER
2360 000346 21CE27A6            MOVE.L A6,USSP        S-STAPELZEIGER
2370 00034A 4E75                RTS
2380                        * ABORT - EINSPRUNG
2390 00034C 21CF27A6     ABRT   MOVE.L A7,USSP  BENUTZER S-STAPELZEIGER
2400 000350 4FF8279E            LEA    SSTACK,A7 MONITOR S-STAPELZEIGER
2410 000354 61D2                BSR    SAVE     REGISTER RETTEN
2420 000356 11FC005B27F7        MOVE.B #$5B,AUS MELDUNG  S
2430                        * HALTEPUNKT BEHANDELN
2440 00035C 303827EA     ABRT1  MOVE.W VPKT,D0  HALTEPUNKT ?
2450 000360 67000008            BEQ    ABRT2    :NEIN
2460 000364 3240                MOVEA  D0,A1    HALTEPUNKTADRESSE
2470 000366 32B827EC            MOVE.W VPKT+2,(A1) ALTEN CODE ZURUECK
2480                        * BEFEHLSZAEHLER UND CODE ANZEIGEN
2490 00036A 41F827A2     ABRT2  LEA    UPC+2,A0  BEFEHLSZAEHLER NACH A0
2510 00036E 3250                MOVE.W (A0),A1  CODE LADEN
2520 000370 6100FE78            BSR    AUSWD    CODE ANZEIGEN
2530 000374 6000FE00            BRA    TAS4     VERARBEITUNGSSCHLEIFE
2540                        * TRAP - EINSPRUNG
2550 000378 21CF27A6     VSTP   MOVE.L A7,USSP  BENUTZER S-STAPELZEIGER
2560 00037C 4FF8279E            LEA    SSTACK,A7 MONITOR S-STAPELZEIGER
2570 000380 61A6                BSR    SAVE     REGISTER RETTEN
2580 000382 11FC00B727F7        MOVE.B #$B7,AUS MELDUNG  .H
2590 000388 55B827A0            SUBQ.L #2,UPC   ADRESSE - 2
2600 00038C 60CE                BRA    ABRT1    HALTEPUNKT UND BEFEHLSZAEHLER
2610                        * TRACE - EINSPRUNG
2620 00038E 21CF27A6     TRAC   MOVE.L A7,USSP  BENUTZER S-STAPELZEIGER
2630 000392 4FF8279E            LEA    SSTACK,A7 MONITOR S-STAPELZEIGER
2640 000396 6190                BSR    SAVE     REGISTER RETTEN
2650 000398 02787FFF27A4        ANDI.W #$7FFF,UCCR TRACE FLAG AUS
2660 00039E 103827F6            MOVE.B MARK,D0  MARKE LADEN
2670 0003A2 6B00FF18            BMI    FUNGO    : NACH HALTEPUNKT WEITER
2680 0003A6 11FC004F27F7        MOVE.B #$4F,AUS MELDUNG  E
2690 0003AC 60BC                BRA    ABRT2
2700                        * FEHLER - EINSPRUNG
2770 0003AE 11FC007E27F6 TAS102 MOVE.B #$7E,MARK CODE 0
2775 0003B4 508F                ADDQ.L #8,A7      KORREKTUR
2780 0003B6 6000005A            BRA    FEHL
2790 0003BA 11FC003027F6 TAS103 MOVE.B #$30,MARK CODE 1
2795 0003C0 508F                ADDQ.L #8,A7      KORREKTUR
2800 0003C2 6000004E            BRA    FEHL
2810 0003C6 11FC006D27F6 TAS104 MOVE.B #$6D,MARK CODE 2
2820 0003CC 60000044            BRA    FEHL
2830 0003D0 11FC007927F6 TAS105 MOVE.B #$79,MARK CODE 3
2840 0003D6 6000003A            BRA    FEHL
2850 0003DA 11FC003327F6 TAS106 MOVE.B #$33,MARK CODE 4
2860 0003E0 60000030            BRA    FEHL
2870 0003E4 11FC005B27F6 TAS107 MOVE.B #$5B,MARK CODE 5
2880 0003EA 60000026            BRA    FEHL
2890 0003EE 11FC005F27F6 TAS108 MOVE.B #$5F,MARK CODE 6
2900 0003F4 6000001C            BRA    FEHL
2910 0003F8 11FC007227F6 TAS109 MOVE.B #$72,MARK CODE 7
2920 0003FE 60000012            BRA    FEHL
2930 000402 11FC007F27F6 TAS110 MOVE.B #$7F,MARK CODE 8
2940 000408 60000008            BRA    FEHL
2950 00040C 11FC007B27F6 TAS111 MOVE.B #$7B,MARK CODE 9
2960 000412 21CF27A6     FEHL   MOVE.L A7,USSP  BENUTZER S-STAPELZEIGER
2970 000416 4FF8279E            LEA    SSTACK,A7 MONITOR S-STAPELZEIGER
2980 00041A 6100FF0C            BSR    SAVE     REGISTER RETTEN
2990 00041E 423827F7            CLR.B  AUS      STATUS LEER
3000 000422 21FC4F05051D
            27F8                MOVE.L #$4F05051D,AUS+1  ERRO
3010 00042A 21FC05000000
            27FC                MOVE.L #$05000000,AUS+5  R
3020 000432 103827F6            MOVE.B MARK,D0
3030 000436 11C027FE            MOVE.B D0,AUS+7  FEHLERCODE ANZEIGEN
3040 00043A 6000FD3A            BRA    TAS4      SCHLEIFE
7000                        * TABELLEN UND KONSTANTEN
7020                        * CODETABELLE BINAER - SIEBENSEGMENT
7030 00043E 7E           COTAB  DC.B   $7E      0
7040 00043F 30                  DC.B   $30      1
7050 000440 6D                  DC.B   $6D      2
7060 000441 79                  DC.B   $79      3
7070 000442 33                  DC.B   $33      4
7080 000443 5B                  DC.B   $5B      5
7090 000444 5F                  DC.B   $5F      6
7100 000445 72                  DC.B   $72      7
7110 000446 7F                  DC.B   $7F      8
7120 000447 7B                  DC.B   $7B      9
7130 000448 77                  DC.B   $77      A
7140 000449 1F                  DC.B   $1F      B
7150 00044A 4E                  DC.B   $4E      C
7160 00044B 3D                  DC.B   $3D      D
7170 00044C 4F                  DC.B   $4F      E
7180 00044D 47                  DC.B   $47      F
7190                        * FUNKTIONSTABELLE
7200 00044E 029A         FUTAB  DC.W   FUNPL    PLUS
7210 000450 02A0                DC.W   FUNMI    MINUS
7220 000452 0288                DC.W   FUNPC    P
7230 000454 02BC                DC.W   FUNGO    G
7240 000456 031A                DC.W   FUNES    E
7250 000458 0276                DC.W   FUNHP    H
7260 00045A 0266                DC.W   FUNDA    D
7270 00045C 0256                DC.W   FUNAD    A
8000                        * BENUTZER - RAM -BEREICH
8010        00002000     URAM   EQU    $2000    BENUTZER-RAM
8020        00002780     USTACK EQU    $2780    BENUTZER-STAPEL
8030                        * MONITOR - RAM -BEREICH
8040        0000279E     SSTACK EQU    $279E    MONITOR-STAPEL
8050        0000279E            ORG    $279E
8055 00279E 00000002     LADD   DS.W   1        LAUFENDE ADRESSE
8060 0027A0 00000004     UPC    DS.L   1        BENUTZER-BEFEHLSZAEHLER
8070 0027A4 00000002     UCCR   DS.W   1        BENUTZER-STATUSREGISTER
8080 0027A6 00000004     USSP   DS.L   1        BENUTZER-STAPELZEIGER
8090 0027AA 00000040     UREG   DS.L   16       BENUTZER-REGISTER
8100 0027EA 00000004     VPKT   DS.L   1        HALTEPUNKT ADRESSE CODE
8110 0027EE 00000002     UINT   DS.W   1        JMP-BEFEHL
8120 0027F0 00000002            DS.W   1        BENUTZER-INTERRUPT-ADRESSE
8130 0027F2 00000002     UTRA   DS.W   1        JMP-BEFEHL
8140 0027F4 00000002            DS.W   1        BENUTZER-TRAP-ADRESSE
8150 0027F6 00000001     MARK   DS.B   1        MARKE/FEHLER-NR.
8160 0027F7 00000009     AUS    DS.B   9        ZIFFERN-ANZEIGE
8170        00002800     REND   EQU    $2800    RAM-ENDE
8180                            END
```

Bild 6. Das Monitorprogramm, erstellt von einem Cross-Assembler

Prof. Dipl.-Ing. Günter Schmitt

Programmbeispiele für 68000-Einplatinencomputer

Eine Einstiegshilfe für „16-Bit-Neulinge“

Angesichts des umfangreichen Registersatzes von 18 Registern und eines 16-Bit-Funktionscodes, der maximal 65 536 verschiedene Befehle zuläßt, erscheint es fast aussichtslos, Programme für den Mikroprozessor 68000 ohne Assemblerhilfe zu erstellen. Um jedoch wenigstens einfache Testprogramme für den ab Seite 65 vorgestellten Computer erstellen zu können, werden hier Listen mit den wichtigsten Befehlen für einen eingeschränkten Registersatz angegeben.

1 Der Registersatz

Bild 1 zeigt die Register des Prozessors 68000; nur die hervorgehobenen Register werden in den Beispielen benutzt. Die acht je 32 Bit langen Datenregister können byteweise, wortweise oder langwortweise angesprochen werden. Dies wird in der Assemblerschreibweise durch den Zusatz .B oder .W oder .L zum Befehl ausgedrückt. Der Befehl MOVE.B D0,D1 überträgt die Bitpositionen 0...7 des Registers D0 in die entsprechenden Bitpositionen des Registers D1, alle anderen Bitpositionen bleiben unberührt.

Die neun je 32 Bit langen Adreßregister können wortweise oder langwortweise angesprochen werden. Sie werden als Indexregister oder als Stapelzeiger verwendet. Zur Adressierung dienen jedoch nur die Bitpositionen 0...23, da der Adreßbus nur aus 23 Anschlüssen besteht.

Das Statusregister besteht aus einem System-Byte und einem Benutzer-Byte. Das System-Byte enthält die drei Interrupt-Masken-Bits I0...I2, das T-Bit für die Einzelschrittsteuerung und das S-Bit, das zwischen dem Supervisor- und dem Anwender-Status unterscheidet. Der Supervisor-Status (S=1) ist für ein Betriebssystem vorgesehen, dem bestimmte (privilegierte) Befehle vorbehalten sind. Das Adreßregister A7 dient als Stapelzeiger bei Unterprogrammen und Ausnahmen und ist doppelt vorhanden; je nachdem, welcher Status eingeschaltet ist. Die Bits des Benutzer-Bytes sind die üblichen Anzeigen für Vorzeichen (N), Null (Z), Überlauf (V) und Übertrag (C). Das X-Bit kennzeichnet einen erweiterten Übertrag ähnlich dem C-Bit.

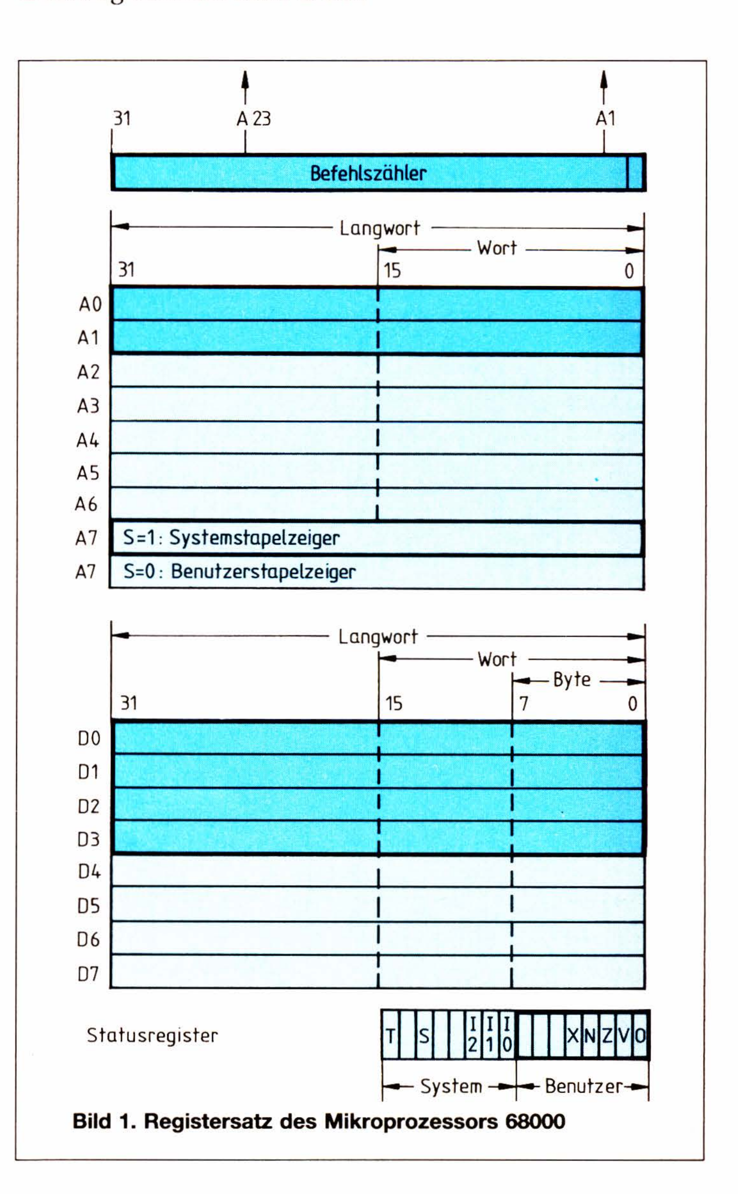


Bild 1. Registersatz des Mikroprozessors 68000

```
 40                             * BILD 2 BEISPIEL FUER STANDARD-ADRESSIERUNGSARTEN
 50 002000 303C1234      START   MOVE.W #$1234,D0   LADE KONSTANTE
 60 002004 3200                  MOVE.W D0,D1       REGISTER - REGISTER
 70 002006 31C02200              MOVE.W D0,$2200    REGISTER - SPEICHER ABSOL
 80 00200A 41F82202              LEA    $2202,A0    ADRESSREGISTER LADEN
 90 00200E 3081                  MOVE.W D1,(A0)     REGISTER - SPEICHER INDIZ
100 002010 60EE                  BRA    START       SCHLEIFE
```

Bild 2. Das Beispiel zeigt die Standard-Adressierungsarten des Prozessors

```
180                           * BILD 4 BEISPIEL FÜR VOR-DEKREMENTIERUNG
200 002022 41F82300                 LEA    $2300,A0    ADRESSREGISTER LADEN
210 002026 313CAAAA           LOOP1 MOVE.W #$AAAA,-(A0) KONSTANTE SPEICHERN
220 00202A B0FC2200                 CMPA   #$2200,A0   ENDADRESSE PRUEFEN
230 00202E 66F6                     BNE    LOOP1       SCHLEIFE
240 002030 4E4F                     TRAP   #15         RUECKSPRUNG MONITOR
```

Bild 4. Beispiel für Vor-Dekrementierung: Jetzt wird das Adreßregister vor dem Datenzugriff dekrementiert

```
110                           * BILD 3 BEISPIEL FUER NACH-INKREMENTIERUNG
130 002012 41F82200                 LEA    $2200,A0    ADRESSREGISTER LADEN
140 002016 30FC5555           LOOP  MOVE.W #$5555,(A0)+ KONSTANTE SPEICHERN
150 00201A B0FC2300                 CMPA   #$2300,A0   ENDADRESSE+2 PRUEFEN
160 00201E 66F6                     BNE    LOOP        SCHLEIFE
170 002020 4E4F                     TRAP   #15         RUECKSPRUNG MONITOR
```

Bild 3. Beispiel für Nach-Inkrementierung: Bei dieser Adressierungsart wird der Inhalt des Adreßregisters nach dem Datenzugriff (vom MOVE-Befehl) automatisch erhöht

```
250                           * BILD 5  BEISPIELE FUER STAPELBEFEHLE
270 002032 43F82500                 LEA    $2500,A1    STAPELZEIGER LADEN
280 002036 303C1234                 MOVE.W #$1234,D0   TESTWERT LADEN
290 00203A 61000004                 BSR    UPRO        UNTERPROGRAMMVERZWEIGUNG
300 00203E 4E4F                     TRAP   #15         RUECKSPRUNG MONITOR
310 002040 2300               UPRO  MOVE.L D0,-(A1)    D0 NACH STAPEL
320 002042 303C5678                 MOVE.W #$5678,D0   TESTWERT LADEN
330 002046 2019                     MOVE.L (A1)+,D0    D0 AUS STAPEL
340 002048 4E75                     RTS                RUECKSPRUNG AUS UNTERPROGR
```

Bild 5. Beispiel für Stapelbefehle: Sie werden durch MOVE-Befehle ersetzt

Tabelle 1. Ausgewählte Befehle des Prozessors 68000

Befehle		Anwendungen
Transfer-		
MOVE		Laden und Speichern von Daten
MOVEQ		Laden und Speichern von Bytekonstanten
LEA		Laden einer Adresse in ein Adreßregister
arithmetische		
ADD		Addieren von dualen Zahlen aus Speicherstellen
ADDI	ADDQ	Addieren von dualen Konstanten
SUB		Subtrahieren von dualen Zahlen aus Speicherstellen
SUBI	SUBQ	Subtrahieren von dualen Konstanten
MULU	MULS	16 Bit × 16 Bit = 32-Bit-Dualzahl MULU ohne Vorzeichen MULS mit Vorzeichen
DIVU	DIVS	32 Bit : 16 Bit = Rest 16 Bit / Quotient 16 Bit DIVU ohne Vorzeichen DIVS mit Vorzeichen
logische		
ANDI		Logisches UND mit einer Konstanten
ORI		Logisches ODER mit einer Konstanten
EORI		Logisches Exklusiv-ODER mit einer Konstanten
Schiebe-		
LSL	LSR	Logisches Links-/Rechtsschieben
ASL	ASR	Arithmetisches Links-/Rechtsschieben
ROL	ROR	Zyklisches Links-/Rechtsschieben
Vergleichs-		
CMP		Vergleich (Testsubtraktion) mit einer Speicherst.
CMPI		Vergleich (Testsubtraktion) mit einer Konstanten
CMPA		Vergleich (Testsubtraktion) im Adreßregister
Verzweigungen		
BRA		Verzweige immer
BEQ	BNE	Verzweige entsprechend dem Z-Bit
BCS	BCC	Verzweige entsprechend dem C-Bit
BMI	BPL	Verzweige entsprechend dem N-Bit
BSR		Verzweige in ein Unterprogramm
Sprünge		
JMP		Springe immer
JMP	( )	Springe immer indiziert
JSR		Springe immer in ein Unterprogramm
RTS		Rücksprung aus einem Unterprogramm
RTE		Rücksprung aus einem Unterbrechungsprogramm
Sonstige		
NOP		No Operation = keine Wirkung
TRAP	#15	Software-Ausnahme (SWI-Befehl des 6800)

2 Der Befehlssatz

Tabelle 1 beschreibt die in den Befehlslisten (*Tabellen 2...5*) enthaltenen Befehle. Der vollständige Befehlssatz steht in den Unterlagen des Herstellers.

Die Mehrzahl der Befehle enthält im Operandenteil zwei Adressen, die durch ein Komma getrennt sind. Ein Beispiel ist der MOVE-Befehl, der gleichermaßen für das Laden und das Speichern von Daten verwendet wird und damit den Befehlen LDA und STA bei manchen 8-Bit-Prozessoren entspricht: MOVE Quelle, Ziel.

Er ist nicht mehr auf Register-Register- oder Register-Speicher-Operationen beschränkt, sondern läßt sich auch zwischen zwei Speicherstellen ausführen. Der Befehl MOVE.W #$5555,$2100 bringt die hexadezimale Wortkonstante 5555 als Direktoperanden ohne Benutzung eines Datenregisters in die adressierte Speicherstelle. Der Befehl MOVE.W (A0),(A1) bringt den Inhalt des durch das Adreßregister A0 adressierten Datenwortes in das durch das Adreßregister A1 adressierte Datenwort. Der Befehl LEA = lade effektive Adresse lädt nicht nur wie der alte Befehl LDX #konst eine Konstante in ein Adreßregister, sondern auch das Ergebnis einer Adreßrechnung.

3 Die Adressierungsarten

Bild 2 faßt die üblichen Standard-Adressierungsarten in einem Testprogramm zusammen. Das Register D0 wird zunächst mit einer Konstanten geladen und anschließend in das Datenregister D1 sowie in die absolut kurz adressierte Speicherstelle $2200 gespeichert. Die beiden nächsten Befehle laden das Adreßregister A0 mit der Adresse einer Speicherstelle und speichern dann den Inhalt von D1 in das dadurch adressierte Speicherwort. Der unbedingte Verzweigungsbefehl enthält im zweiten Byte des Befehlswortes den Abstand zum Sprungziel, der wie üblich in Bytes gezählt und im Zweierkomplement eingesetzt wird.

Bei der Adressierung von Bereichen wie z. B. Tabellen muß das zur Adressierung benutzte Register jeweils um die Schrittweite erhöht bzw. vermindert werden. Die Adressierungsart Nach-Inkrement erhöht den Inhalt des Adreßregisters nach dem Datenzugriff um die Zahl der verwendeten Bytes, nach einem Wortzugriff also um die

Schrittweite 2. *Bild 3* zeigt ein Testprogramm, das einen Speicherbereich in aufsteigender Folge mit einer Konstanten füllt. *Bild 4* zeigt ein ähnliches Programm, jedoch in absteigender Folge mit der Adressierungsart Vor-Dekrement, bei der das Adreßregister vor dem Datenzugriff vermindert wird. MOVE-Befehle der beiden genannten Adressierungsarten ersetzen die nicht vorhandenen PUSH- und PULL-Befehle zum Schreiben und Lesen des Stapels. *Bild 5* zeigt als Beispiel ein Hauptprogramm, das das Register A1 als Stapelzeiger mit einer Anfangsadresse lädt, und ein Unterprogramm, das das Register D0 in den Stapel rettet und wieder zurücklädt. Jedes Adreßregister kann als Stapelzeiger verwendet werden; damit lassen sich insgesamt acht Stapel anlegen. Unterprogrammaufrufe und Ausnahmen (z. B. Interrupt) arbeiten mit dem Adreßregister A7.

Die Verzweigungsbefehle, bei denen die Adresse des Sprungziels durch Addition eines Abstandes zum Befehlszählerstand errechnet wird, haben gegenüber den Sprungbefehlen mit absoluter Adressierung den Vorteil, daß das Programm unabhängig davon ist, an welcher Stelle des Speichers es steht. Die relative Adressierung der Verzweigungsbefehle wird beim 68000 auf + 32 768 Bytes ausgedehnt. Reicht das zweite Byte des Befehlswortes zur Aufnahme der Sprungweite nicht aus, so wird es auf Null gesetzt, und die Sprungweite wird im folgenden Wort als vorzeichenbehaftete Dualzahl abgelegt. Ein Beispiel ist der Unterprogrammaufruf von Bild 5. Der Cross-Assembler setzt bei allen Vorwärtssprüngen automatisch die lange Sprungweite ein.

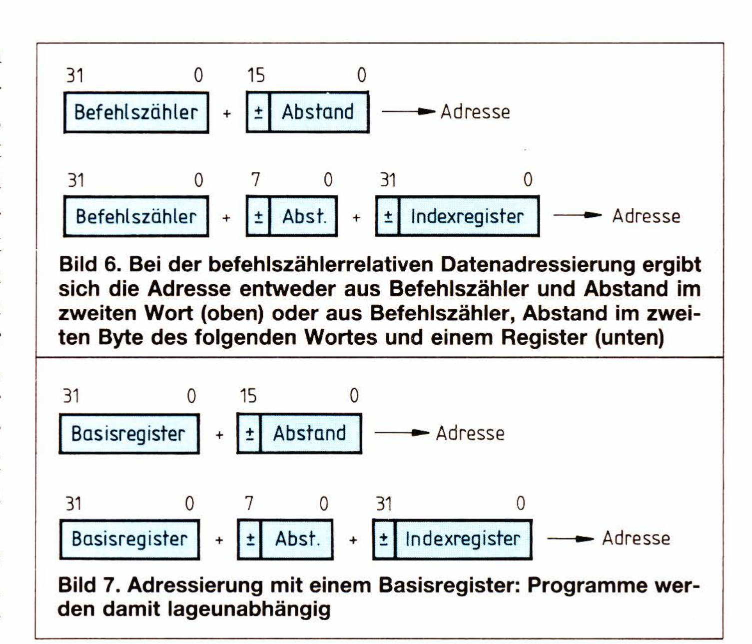


Bild 6. Bei der befehlszählerrelativen Datenadressierung ergibt sich die Adresse entweder aus Befehlszähler und Abstand im zweiten Wort (oben) oder aus Befehlszähler, Abstand im zweiten Byte des folgenden Wortes und einem Register (unten)

Bild 7. Adressierung mit einem Basisregister: Programme werden damit lageunabhängig

Die relative Adressierung läßt sich beim 68000 auch auf Datenadressen anwenden. *Bild 6* demonstriert die beiden Möglichkeiten. Die Datenadresse wird gebildet aus dem Befehlszähler und einem Abstand im zweiten Wort bzw. aus dem Befehlszähler, einem Abstand im zweiten Byte des folgenden Wortes und dem Inhalt eines Registers, das als Indexregister für die Adressierung von Bereichen verwendet werden kann. Damit ist es möglich, Programme zu schreiben, die sowohl von

Tabelle 2. MOVE-Befehle

	D0			D1			A0			A1			(A0)			(A1)			(A0)+			(A1)+			-(A0)			-(A1)			-(A7)			adr		
	Code	W	Z	Code	W	Z	Code	W	Z	Code	W	Z	Code	W	Z	Code	W	Z	Code	W	Z	Code	W	Z	Code	W	Z	Code	W	Z	Code	W	Z	Code	W	Z
MOVE.B D0,				1200	1								1080	1		1280	1		10C0	1		12C0	1		1100	1		1300	1		1F00	1		11C0	2	
MOVE.W D0,				3200	1		3040	1		3240	1		3080	1		3280	1		30C0	1		32C0	1		3100	1		3300	1		3F00	1		31C0	2	
MOVE.L D0,				2200	1		2040	1		2240	1		2080	1		2280	1		20C0	1		22C0	1		2100	1		2300	1		2F00	1		21C0	2	
MOVE.B D1,	1001	1											1081	1		1281	1		10C1	1		12C1	1		1101	1		1301	1		1F01	1		11C1	2	
MOVE.W D1,	3001	1					3041	1		3241	1		3081	1		3281	1		30C1	1		32C1	1		3101	1		3301	1		3F01	1		31C1	2	
MOVE.L D1,	2001	1					2041	1		2241	1		2081	1		2281	1		20C1	1		22C1	1		2101	1		2301	1		2F01	1		21C1	2	
MOVE.W A0,	3008	1		3208	1					3248	1		3088	1		3288	1		30C8	1		32C8	1		3108	1		3308	1		3F08	1		31C8	2	
MOVE.L A0,	2008	1		2208	1					2248	1		2088	1		2288	1		20C8	1		22C8	1		2108	1		2308	1		2F08	1		21C8	2	
MOVE.W A1,	3009	1		3209	1		3049	1					3089	1		3289	1		30C9	1		32C9	1		3109	1		3309	1		3F09	1		31C9	2	
MOVE.L A1,	2009	1		2209	1		2049	1					2089	1		2289	1		20C9	1		22C9	1		2109	1		2309	1		2F09	1		21C9	2	
MOVE.B #b,	103C	2		123C	2								10BC	2		12BC	2		10FC	2		12FC	2		113C	2		133C	1		1F3C	2		11FC	3	
MOVE.W #bb,	303C	2		323C	2		307C	2		327C	2		30BC	2		32BC	2		30FC	2		32FC	2		313C	2		333C	2		3F3C	2		31FC	3	
MOVE.L #bbbb,	203C	3		223C	3		207C	3		227C	3		20BC	3		22BC	3		20FC	3		22FC	3		213C	3		233C	3		2F3C	3		21FC	4	
MOVE.B (A0),	1010	1		1210	1											1290	1					12D0	1					1310	1		1F10	1		11D0	2	
MOVE.W (A0),	3010	1		3210	1		3050	1		3250	1					3290	1					32D0	1					3310	1		3F10	1		31D0	2	
MOVE.L (A0),	2010	1		2210	1		2050	1		2250	1					2290	1					22D0	1					2310	1		2F10	1		21D0	2	
MOVE.B (A1),	1011	1		1211	1								1091	1					10D1	1					1111	1					1F11	1		11D1	2	
MOVE.W (A1),	3011	1		3211	1		3051	1		3251	1		3091	1					30D1	1					3111	1					3F11	1		31D1	2	
MOVE.L (A1),	2011	1		2211	1		2051	1		2251	1		2091	1					20D1	1					2111	1					2F11	1		21D1	2	
MOVE.B (A0)+,	1018	1		1218	1											1298	1					12D8	1					1318	1		1F18	1		11D8	2	
MOVE.W (A0)+,	3018	1		3218	1		3058	1		3258	1					3298	1					32D8	1					3318	1		3F18	1		31D8	2	
MOVE.L (A0)+,	2018	1		2218	1		2058	1		2258	1					2298	1					22D8	1					2318	1		2F18	1		21D8	2	
MOVE.B (A1)+,	1019	1		1219	1								1099	1					10D9	1					1119	1					1F19	1		11D9	2	
MOVE.W (A1)+,	3019	1		3219	1		3059	1		3259	1		3099	1					30D9	1					3119	1					3F19	1		31D9	2	
MOVE.L (A1)+,	2019	1		2219	1		2059	1		2259	1		2099	1					20D9	1					2119	1					2F19	1		21D9	2	
MOVE.B -(A0),	1020	1		1220	1											12A0	1					12E0	1					1320	1		1F20	1		11E0	2	
MOVE.W -(A0),	3020	1		3220	1		3060	1		3260	1					32A0	1					32E0	1					3320	1		3F20	1		31E0	2	
MOVE.L -(A0),	2020	1		2220	1		2060	1		2260	1					22A0	1					22E0	1					2320	1		2F20	1		21E0	2	
MOVE.B -(A1),	1021	1		1221	1								10A1	1					10E1	1					1121	1					1F21	1		11E1	2	
MOVE.W -(A1),	3021	1		3221	1		3061	1		3261	1		30A1	1					30E1	1					3121	1					3F21	1		31E1	2	
MOVE.L -(A1),	2021	1		2221	1		2061	1		2261	1		20A1	1					20E1	1					2121	1					2F21	1		21E1	2	
MOVE.B (A7)+,	101F	1		121F	1								109F	1		129F	1		10DF	1		12DF	1		111F	1		131F	1					11DF	2	
MOVE.W (A7)+,	301F	1		321F	1		305F	1		325F	1		309F	1		329F	1		30DF	1		32DF	1		311F	1		331F	1					31DF	2	
MOVE.L (A7)+,	201F	1		221F	1		205F	1		225F	1		209F	1		229F	1		20DF	1		22DF	1		211F	1		231F	1					21DF	2	
MOVE.B adr,	1038	2		1238	2								10B8	2		12B8	2		10F8	2		12F8	2		1138	2		1338	2		1F38	2				
MOVE.W adr,	3038	2		3238	2		3078	2		3278	2		30B8	2		32B8	2		30F8	2		32F8	2		3238	2		3338	2		3F38	2				
MOVE.L adr,	2038	2		2238	2		2078	2		2278	2		20B8	2		22B8	2		20F8	2		22F8	2		2138	2		2338	2		2F38	2				

der Sprungadressierung als auch von der Datenadressierung her lageunabhängig sind.

Bild 7 veranschaulicht die Möglichkeit, mit Hilfe eines Adreßregisters als Basisregister ebenfalls eine lageunabhängige Adressierung vorzunehmen. Zum Inhalt eines Adreßregisters wird ein Abstand bzw. ein Abstand plus Inhalt eines weiteren Registers (als Indexregister) addiert. Das Adreßregister wird beim Start des Programms geladen und enthält die Basis der Datenadressierung.

4 Befehlslisten

Die Adressierungsarten „Relativ“ und „Basisregister“ fehlen in Tabelle 1. Tabelle 2 enthält MOVE-Befehle. In der Spalte Code steht das Befehlswort, die Spalte W gibt die Zahl der Wörter des Befehls an. Bei allen MOVE-Befehlen werden mit Ausnahme des Ladens eines Adreßregisters das N- und das Z-Bit des Bedingungsregisters verändert; das V- und das C-Bit werden gelöscht. Tabelle 3 faßt arithmetische und logische Befehle für die Register D0...D3 zusammen. In der Spalte Bedingungen bedeutet ein x, daß das Bit entsprechend dem Ergebnis verändert wird; eine 0 bedeutet, daß das Bit gelöscht wird. Ist die Spalte leer, so findet keine Änderung statt. Tabelle 4 faßt die Schiebebefehle für Verschiebungen um maximal acht Bit zusammen. Entsprechend den rechts angeordneten Tabellen ist die Zahl der Verschiebungen in das Befehlswort einzutragen. Bei den Verzweigungsbefehlen (Tab. 5) ist in das 2. Byte des Befehlswortes der Abstand zum Sprungziel einzutragen.

5 Programmbeispiele

Bild 8 enthält eine Reihe von einfachen Programmbeispielen, die so gestaltet wurden, daß sie mit den Befehlstabellen übersetzt werden konnten. Dadurch mußten einige umständliche Programmschritte in Kauf genommen werden, die bei der Benutzung eines Assemblers unter Verwendung des vollen Befehls- und Registersatzes entfallen würden. Die Speicherstellen $2100 bis $210D enthalten Adressen bzw. Daten, die vor dem Start der Programme mit Hilfe des Monitors eingegeben werden müssen. Die Programme werden mit dem Befehl TRAP#15 abgeschlossen, der wie ein Haltepunkt wirkt und in den Monitor zurückspringt.

Das Speichertestprogramm läuft in einer unendlichen Schleife und speichert bei einem Fehler die Adresse des fehlerhaften Speicherwortes in die Speicherstelle ZIEL = $2004.

Das Verschiebeprogramm verschiebt einen Speicherbereich an eine neue Stelle. Dabei dürfen sich beide Bereiche nur überlappen, wenn die Zieladresse kleiner als die Anfangsadresse ist.

Tabelle 3. Arithmetische und logische Befehle

	Wirkung	D0 Code	D0 W	D0 Z	D1 Code	D1 W	D1 Z	D2 Code	D2 W	D2 Z	D3 Code	D3 W	D3 Z	Bedingung ...X	N	Z	V	C
MOVE.L D0,	D0 ⇒ rg	2000	1		2200	1		2400	1		2600	1			X	X	0	0
MOVE.L D1,	D1 ⇒ rg	2001	1		2201	1		2401	1		2601	1			X	X	0	0
MOVE.L D2,	D2 ⇒ rg	2002	1		2202	1		2402	1		2602	1			X	X	0	0
MOVE.L D3,	D3 ⇒ rg	2003	1		2203	1		2403	1		2603	1			X	X	0	0
MOVEQ #xx,	Byte⇒ rg	70xx	1		72xx	1		74xx	1		76xx	1			X	X	0	0
ADDQ.L #1,	rg+1⇒ rg	5280	1		5281	1		5282	1		5283	1		X	X	X	X	X
SUBQ.L #1,	rg-1⇒ rg	5380	1		5381	1		5382	1		5383	1		X	X	X	X	X
ADD.B D0,	rg+D0⇒rg	D000	1		D200	1		D400	1		D600	1		X	X	X	X	X
ADD.W D0,	rg+D0⇒rg	D040	1		D240	1		D440	1		D640	1		X	X	X	X	X
ADD.B D1,	rg+D1⇒rg	D001	1		D201	1		D401	1		D601	1		X	X	X	X	X
ADD.W D1,	rg+D1⇒rg	D041	1		D241	1		D441	1		D641	1		X	X	X	X	X
ADDI.B #b,	rg+ b⇒rg	0600	2		0601	2		0602	2		0603	2		X	X	X	X	X
ADDI.W #bb,	rg+bb⇒rg	0640	2		0641	2		0642	2		0643	2		X	X	X	X	X
SUB.B D0,	rg-D0⇒rg	9000	1		9200	1		9400	1		9600	1		X	X	X	X	X
SUB.W D0,	rg-D0⇒rg	9040	1		9240	1		9440	1		9640	1		X	X	X	X	X
SUB.B D1,	rg-D1⇒rg	9001	1		9201	1		9401	1		9601	1		X	X	X	X	X
SUB.W D1,	rg-D1⇒rg	9041	1		9241	1		9441	1		9641	1		X	X	X	X	X
SUBI.B #b,	rg- b⇒rg	0400	2		0401	2		0402	2		0403	2		X	X	X	X	X
SUBI.W #bb,	rg-bb⇒rg	0440	2		0441	2		0442	2		0443	2		X	X	X	X	X
CMP.B D0,	rg - D0	B000	1		B200	1		B400	1		B600	1			X	X	X	X
CMP.W D0,	rg - D0	B040	1		B240	1		B440	1		B640	1			X	X	X	X
CMP.B D1,	rg - D1	B001	1		B201	1		B401	1		B601	1			X	X	X	X
CMP.W D1,	rg - D1	B041	1		B241	1		B441	1		B641	1			X	X	X	X
CMPI.B #b,	rg - b	0C00	2		0C01	2		0C02	2		0C03	2			X	X	X	X
CMPI.W #bb,	rg - bb	0C40	2		0C41	2		0C42	2		0C43	2			X	X	X	X
CMPI.L #bbbb,	rg-bbbb	0C80	3		0C81	3		0C82	3		0C83	3			X	X	X	X
ANDI.B #b,	rg AND #	0200	2		0201	2		0202	2		0203	2			X	X	0	0
ANDI.W #bb,	rg AND #	0240	2		0241	2		0342	2		0243	2			X	X	0	0
ANDI.L #bbbb,	rg AND #	0280	3		0281	3		0282	3		0283	3			X	X	0	0
ORI.B #b,	rg OR #	0000	2		0001	2		0002	2		0003	2			X	X	0	0
ORI.W #bb,	rg OR #	0040	2		0041	2		0042	2		0043	2			X	X	0	0
ORI.L #bbbb,	rg OR #	0080	3		0081	3		0082	3		0083	3			X	X	0	0
EORI.B #b,	rg XOR #	0A00	2		0A01	2		0A02	2		0A03	2			X	X	0	0
EORI.W #bb,	rg XOR #	0A40	2		0A41	2		0A42	2		0A43	2			X	X	0	0
EORI.L #bbbb,	rg XOR #	0A80	3		0A81	3		0A82	3		0A83	3			X	X	0	0
MULU D0,	rg*D0⇒rg	C0C0	1		C2C0	1		C4C0	1		C6C0	1			X	X	0	0
MULU D1,	rg*D1⇒rg	C0C1	1		C2C1	1		C4C1	1		C6C1	1			X	X	0	0
MULU #bb,	rg*bb⇒rg	C0FC	2		C2FC	2		C4FC	2		C6FC	2			X	X	0	0
MULS D0,	rg*D0⇒rg	C1C0	1		C3C0	1		C5C0	1		C7C0	1			X	X	0	0
MULS D1,	rg*D1⇒rg	C1C1	1		C3C1	1		C5C1	1		C7C1	1			X	X	0	0
MULS #bb,	rg*bb⇒rg	C1FC	2		C3FC	2		C5FC	2		C7FC	2			X	X	0	0
DIVU D0,	rg/D0⇒rg	80C0	1		82C0	1		84C0	1		86C0	1			X	X	X	0
DIVU D1,	rg/D1⇒rg	80C1	1		82C1	1		84C1	1		86C1	1			X	X	X	0
DIVU #bb,	rg/bb⇒rg	80FC	2		82FC	2		84FC	2		86FC	2			X	X	X	0
DIVS D0,	rg/D0⇒rg	81C0	1		83C0	1		85C0	1		87C0	1			X	X	X	0
DIVS D1,	rg/D1⇒rg	81C1	1		83C1	1		85C1	1		87C1	1			X	X	X	0
DIVS #bb,	rg/bb⇒rg	81FC	2		83FC	2		85FC	2		87FC	2			X	X	X	0

Tabelle 4. Schiebebefehle

	Wirkung	D0 Code	D0 W	D0 Z	D1 Code	D1 W	D1 Z	Bedingung ...X	N	Z	V	C
LSL.B #n,	C ← [←] ← 0	Ex08	1		Ex09	1		X	X	X	0	X
LSL.W #n,	X ←	Ex48	1		Ex49	1		X	X	X	0	X
LSL.L #n,		Ex88	1		Ex89	1		X	X	X	0	X
ASL.B #n,	C ← [←] ← 0	Ex00	1		Ex01	1		X	X	X	0	X
ASL.W #n,	X ←	Ex40	1		Ex41	1		X	X	X	0	X
ASL.L #n,		Ex80	1		Ex81	1		X	X	X	0	X
ROL.B #n,	C ← [←]	Ex18	1		Ex19	1			X	X	0	X
ROL.W #n,		Ex58	1		Ex59	1			X	X	0	X
ROL.L #n,		Ex98	1		Ex99	1			X	X	0	X
LSR.B #n,	0 → [→] → C	Ey08	1		Ey09	1		X	X	X	0	X
LSR.W #n,	→ X	Ey48	1		Ey49	1		X	X	X	0	X
LSR.L #n,		Ey88	1		Ey89	1		X	X	X	0	X
ASR.B #n,	[Vz →] → C	Ey00	1		Ey01	1		X	X	X	0	X
ASR.W #n,	→ X	Ey40	1		Ey41	1		X	X	X	0	X
ASR.L #n,		Ey80	1		Ey81	1		X	X	X	0	X
ROR.B #n,	[→] → C	Ey18	1		Ey19	1			X	X	0	X
ROR.W #n,		Ey58	1		Ey59	1			X	X	0	X
ROR.L #n,		Ey98	1		Ey99	1			X	X	0	X

Tabelle

n	x
1	3
2	5
3	7
4	9
5	B
6	D
7	F
8	1

n	y
1	2
2	4
3	6
4	8
5	A
6	C
7	E
8	0

Tabelle 5. Verzweigungsbefehle

	Wirkung	Code	W	Z	Bedingung ...X	N	Z	V	C
BRA adresse	verzweige immer	60dd	1						
BEQ adresse	verzweige Z = 1	67dd	1						
BNE adresse	verzweige Z = 0	66dd	1						
BCS adresse	verzweige C = 1	65dd	1						
BCC adresse	verzweige C = 0	64dd	1						
BMI adresse	verzweige N = 1	6Bdd	1						
BPL adresse	verzweige N = 0	6Add	1						
BSR adresse	verzweige Upro.	61dd	1						
JMP adresse	springe immer	4EF8	2						
JMP (A0)	springe immer	4ED0	1						
JMP (A1)	springe immer	4ED1	1						
JSR adresse	springe Unterpr.	4EB8	2						
JSR (A0)	springe Unterpr.	4E90	1						
JSR (A1)	springe Unterpr.	4E91	1						
RTS	Rückspr. Upro	4E75	1						
RTE	Rückspr. Ausnahme	4E73	1						
NOP	keine	4E71	1						
TRAP #0	Software-Ausnahme	4E40	1						
TRAP #15	Software-Ausnahme	4E4F	1						
LEA adr,A0	lade Adresse ⇒ A0	41F8	2						
LEA adr,A1	lade Adresse ⇒ A1	43F8	2						
LEA adr,A7	lade Adresse ⇒ A7	4FF8	2						
CMPA #adr,A0	A0 - Konstante	B0FC	2			X	X	X	X
CMPA #adr,A1	A1 - Konstante	B2FC	2			X	X	X	X
CMPA #adr,A7	A7 - Konstante	BEFC	2			X	X	X	X
CMPA adr,A0	A0 - Wort	B0F8	2			X	X	X	X
CMPA adr,A1	A1 - Wort	B2F8	2			X	X	X	X
CMPA adr,A7	A7 - Wort	BEF8	2			X	X	X	X

```
                        * BILD 8 ALLGEMEINE BEISPIELPROGRAMME
                        * DATENBEREICH
        00002100                ORG     $2100
002100  00000002        ANF     DS.W    1               ANFANGSADRESSE
002102  00000002        END     DS.W    1               ENDADRESSE+2
002104  00000002        ZIEL    DS.W    1               ZIELADRESSE
002106  00000004        DEZI    DS.L    1               DEZIMALZAHL ALS ASCII-ZEI
00210A  00000004        DUAL    DS.L    1               DUALZAHL
                        * SPEICHERTEST
        00002110                ORG     $2110
002110  30782100        TEST    MOVE.W  ANF,A0          ANFANGSADRESSE LADEN
002114  3010            LOOP2   MOVE.W  (A0),D0         WERT RETTEN
002116  30BC5555                MOVE.W  #$5555,(A0)     TESTWERT SPEICHERN
00211A  3210                    MOVE.W  (A0),D1         TESTWERT ZURÜCKLADEN
00211C  0C415555                CMPI.W  #$5555,D1       WERT PRUEFEN
002120  6600000C                BNE     ERROR           BEI UNGLEICH FEHLER
002124  30C0                    MOVE.W  D0,(A0)+        ALTEN WERT ZURUECK
002126  B0F82102                CMPA.W  END,A0          ENDADRESSE PRUEFEN
00212A  65E8                    BCS     LOOP2           WEITER PRUEFEN
00212C  60E2                    BRA     TEST            NEUER DURCHLAUF
00212E  3080            ERROR   MOVE.W  D0,(A0)         ALTEN WERT ZURUECK
002130  31C82104                MOVE.W  A0,ZIEL         FEHLERADRESSE SPEICHERN
002134  4E4F                    TRAP    #15             RUECKSPRUNG MONITOR
                        * SPEICHERBEREICH VERSCHIEBEN
002136  30782100        VERS    MOVE.W  ANF,A0          ANFANGSADRESSE LADEN
00213A  32782104                MOVE.W  ZIEL,A1         ZIELADRESSE LADEN
00213E  32D8            LOOP3   MOVE.W  (A0)+,(A1)+     UMSPEICHERN
002140  B0F82102                CMPA.W  END,A0          ENDADRESSE+2 PRUEFEN
002144  65F8                    BCS     LOOP3           SCHLEIFE
002146  4E4F                    TRAP    #15             RUECKSPRUNG MONITOR
                        * DEZIMAL (ASCII) - DUAL - UMWANDLUNG
002148  20382106        DEDU    MOVE.L  DEZI,D0         DEZIMALZAHL LADEN
00214C  7200                    MOVEQ   #0,D1           HILFSREGISTER
00214E  7400                    MOVEQ   #0,D2           DUALWERT
002150  7604                    MOVEQ   #4,D3           DURCHLAUFZAEHLER
002152  C4FC000A        LOOP4   MULU    #10,D2          MAL 10
002156  E198                    ROL.L   #8,D0           STELLE SCHIEBEN
002158  0200000F                ANDI.B  #$0F,D0         ASCII UMWANDELN
00215C  1200                    MOVE.B  D0,D1           HILFSREGISTER
00215E  D441                    ADD.W   D1,D2           STELLE ADDIEREN
002160  5343                    SUBQ    #1,D3           ZAEHLER - 1
002162  66EE                    BNE     LOOP4           WEITER
002164  2002                    MOVE.L  D2,D0           UMLADEN
002166  21C0210A                MOVE.L  D0,DUAL         DUALWERT SPEICHERN
00216A  4E4F                    TRAP    #15             RUECKSPRUNG MONITOR
                        * DUAL - DEZIMAL (ASCII) - UMWANDLUNG
00216C  2038210A        DUDEZ   MOVE.L  DUAL,D0         DUALZAHL LADEN
002170  2400                    MOVE.L  D0,D2           UMSPEICHERN
002172  7604                    MOVEQ   #4,D3           DURCHLAUFZAEHLER
002174  84FC000A        LOOP5   DIVU    #10,D2          DURCH 10
002178  2002                    MOVE.L  D2,D0           NACH HILFSREGISTER
00217A  02820000FFFF            ANDI.L  #$0000FFFF,D2 REST MASKIEREN
002180  E088                    LSR.L   #8,D0           REST 8 BIT NACH RECHTS
002182  E088                    LSR.L   #8,D0           REST 8 BIT NACH RECHTS
002184  00000030                ORI.B   #$30,D0         $30 DAZU
002188  1200                    MOVE.B  D0,D1           ZAHL VERSCHIEBEN
00218A  E099                    ROR.L   #8,D1           ZAHL EINFUEGEN
00218C  5343                    SUBQ    #1,D3           ZAEHLER - 1
00218E  66E4                    BNE     LOOP5           WEITER
002190  21C12106                MOVE.L  D1,DEZI         DEZIMALZAHL SPEICHERN
002194  4E4F                    TRAP    #15             RUECKSPRUNG MONITOR
```

Bild 8. Allgemeine Beispielprogramme

```
                        * BILD 9  UNTERPROGRAMMAUFRUF TERMINALMONITOR
                        * BEISPIEL TEXT SPEICHERN UND DRUCKEN
002196  41F82200        TEIN    LEA     $2200,A0        ANFANGSADRESSE LADEN
00219A  4EB801C8        LOOP6   JSR     $01C8           EINE = ZEICHEN LESEN
00219E  4EB80100                JSR     $0100           AUS = ZEICHEN IM ECHO AUS
0021A2  10C0                    MOVE.B  D0,(A0)+        ZEICHEN ABLEGEN
0021A4  0C000004                CMPI.B  #$04,D0         AUF ENDE-ZEICHEN PRUEFEN
0021A8  66F0                    BNE     LOOP6           WEITER
0021AA  4E4F                    TRAP    #15             RUECKSPRUNG MONITOR
                        * TEXT AUSGEBEN
0021AC  41F82200        TAUS    LEA     $2200,A0        ANFANGSADRESSE LADEN
0021B0  1018            LOOP7   MOVE.B  (A0)+,D0        ZEICHEN LADEN
0021B2  0C000004                CMPI.B  #$04,D0         ENDEZEICHEN ?
0021B6  67000008                BEQ     FINA            TEXT FERTIG: ENDE
0021BA  4EB80100                JSR     $0100           ZEICHEN AUSGEBEN
0021BE  60F0                    BRA     LOOP7           WEITER
0021C0  4E4F            FINA    TRAP    #15             RUECKSPRUNG MONITOR
```

Bild 9. Aufruf eines Terminal-Unterprogramms

Das Programm zur Dezimal-/Dual-Umwandlung verlangt die Eingabe von Dezimalzahlen als ASCII-Zeichen, die in einem Byte codiert sein müssen; eine Prüfung der Codierung findet nicht statt.

Das Programm zur Dual-/Dezimal-Umwandlung verlangt die Eingabe einer Dualzahl kleiner 9999; auch hier findet keine Prüfung des Zahlenbereiches statt.

Die *Tabelle 6* enthält eine Zusammenfassung der Unterprogramme des Terminal-Monitors, die der Benutzer in seinem Programm aufrufen kann. *Bild 9* zeigt dazu ein Beispielprogramm, mit dem ein beliebiger Text, den man vom Terminal aus eingibt, im Speicher abgelegt wird. Das ASCII-Steuerzeichen EOT beendet die Eingabe. Das folgende Ausgabeprogramm gibt den Text wieder aus; es kann beliebig oft aufgerufen werden.

Die *Tabelle 7* faßt die Unterprogramme des Tasten-Monitors mit den entsprechenden RAM-Ausgabeadressen und Codetabellen zusammen.

Tabelle 6: Unterprogramme des Terminal-Monitors

Name	Adresse	Aufgabe
AUS	$0100	1 Zeichen aus D0 ausgeben D7 zerstört
AUSB4	$010E	1 Leerzeichen und 4 Zeichen aus D0 ausgeben D2, D7 zerstört
AUS4	$0118	4 Zeichen aus D0 ausgeben D2, D7 zerstört
AUSWD	$0124	1 Leerzeichen und 1 Wort codiert aus D0 ausgeben D0, D1, D2, D7 zerstört
AUSDW	$0150	1 Leerzeichen und 1 Doppelwort codiert aus D0 ausgeben D0, D1, D2, D7 zerstört
EIN	$015A	1 Zeichen vom Terminal nach D0 übertragen D0 zerstört
EINE	$01B8	1 Zeichen vom Terminal nach D0 übertragen und im Echo ausgeben D0 zerstört
EIN4	$01DA	4 Zeichen bis Abbruch vom Terminal nach D0 übertragen und im Echo ausgeben D0 zerstört
DEC0	$020E	4 Zeichen aus D0 nach D1 dual umwandeln C = 1 : Fehler D0, D1, D2, D3, D4 zerstört

Tabelle 7. Unterprogramme des Tasten-Monitors

Name	Adresse	Aufgabe
ANZ	$0186	RAM-Bereich $27F7...27FF anzeigen. A6,D6,D7 zerstört
TEST	$01AA	Tastatur prüfen C=0: keine Taste C=1: D0 = Code, D1 = Zeile D0 und D1 zerstört
UMCO	$01D0	duales Wort aus D0 nach 7-Segmentcode in D1 umcodieren A6,D6,D7 zerstört
AUSWD	$01EA	Wort aus (A1) umcodieren und nach $27FC bringen A6,D7,D6,D1,D0 zerstört
AUSAD	$01F2	Wort aus (A0) umcodieren und nach $27F8 bringen A6,D7,D6,D1,D0 zerstört
WAIT	$01FA	warten auf Tastenfreigabe mit Anzeige. D0,D1 zerstört
COTAB	$043A	Tabelle mit 7-Segment-Code von 0...F

Andrew Barth

32-Bit-Prozessor ersetzt 8-Bit-CPU

An einem praktischen Beispiel wird gezeigt, wie einfach es ist, mit dem 32-Bit-Prozessor MC68008 einen 8-Bit-CPU-Baustein in einem existierenden System zu ersetzen. Auf diese Weise ist es möglich, ohne zusätzlichen Entwicklungsaufwand zu einem Mikrocomputer höherer Leistung zu gelangen. Als typischer Vertreter existierender 8-Bit-Systeme wurde das „Micromodule 19" (M68MM19A) gewählt.

Das Einplatinen-System M68MM19A besitzt eine 2-MHz-MPU MC6809, 16 KByte ROM/EPROM, 2 KByte statisches RAM, ein asynchrones serielles Datenport (ACIA MC68B50) mit RS-232C-/RS-422-/RS-423-Interface, ein paralleles Drucker-Interface (PIA MC68B21), drei 16-Bit-Zähler (PTM MC68M40) sowie externe Adressen-, Daten- und Steuer-Bus-Puffer.

Die hier beschriebene Interface-Logik für den MC68008 wurde speziell für die Anpassung an die Platine M68MM19A entwickelt, obwohl sie prinzipiell mit allen Systemen, die auf dem MC6809 basieren, arbeiten kann. Eine Bus-Arbitrationslogik wurde nicht vorgesehen, diese ist allerdings erforderlich, wenn ein dynamischer Speicher oder zusätzliche potentielle Bus-Master (z. B. DMA-Bausteine) benutzt werden sollen.

1 MPU-Baustein MC68008

Der in [1] bereits ausführlich beschriebene MPU-Baustein MC68008 ist Mitglied der Mikroprozessorfamilie M68000. Dieser Baustein besitzt die gleiche interne Architektur wie die MPU MC68000 und ist vollständig mit ihr softwarekompatibel. Im Gegensatz dazu besitzt der Typ MC68008 einen externen 8-Bit-Datenbus. Aus diesem Grunde kann man mit der MPU kosteneffektive Systeme aufbauen, die einen 8-Bit-Bus benutzen, ohne daß auf die Vorteile einer 32-Bit-Mikroprozessorarchitektur verzichtet werden muß (*Bild 1*).

Die MPU MC68008 bietet dem Benutzer 17 Register für allgemeine Verwendung mit jeweils 32 Bit, 56 grundlegende Befehlstypen sowie 14 Grund-Adressierungsarten mit vielen untergeordneten Arten (*Bild 2*). Die Kombination von Befehls- und Datentypen sowie Adressierungsarten ergeben einige Tausend nützliche Befehle. Ein nichtsegmentierter linearer Adreßbereich von 1 MByte erlaubt, daß große modulare Programme entwickelt und ausgeführt werden können. An Speicher- und Peripherie-Einheiten wird der MC68008 über Hochleistungs-Daten- und Adreß-Busse (kein Multiplex) angeschlossen.

2 Interface zwischen MC68008 und Speicher bzw. Peripherieeinheiten

Im Normalbetrieb führt die MPU MC68008 Datentransfers zum oder vom Speicher bzw. Peripherieeinheiten in asynchroner Betriebsart durch. Handshake-Steuerleitungen, Adreß-Strobe (AS), Daten-Strobe (DS), Schreib/Lese-Leitung (RW) sowie das Data-Transfer-Acknowledge-Signal (DTACK) ermöglichen die optimale Anpassung der Datentransferrate an die einzelne Einheit, auf die zugegriffen werden soll. Nach Beginn eines Buszyklus teilt die Peripherieeinheit durch Ausgabe des DTACK-Signals der MPU mit, daß sie bereit ist, Daten zu empfangen oder auszusenden. Die zeitliche Lage des DTACK-Signals innerhalb eines gegebenen Bus-Zyklus ist auf die Datenzugriffszeit der Einheit

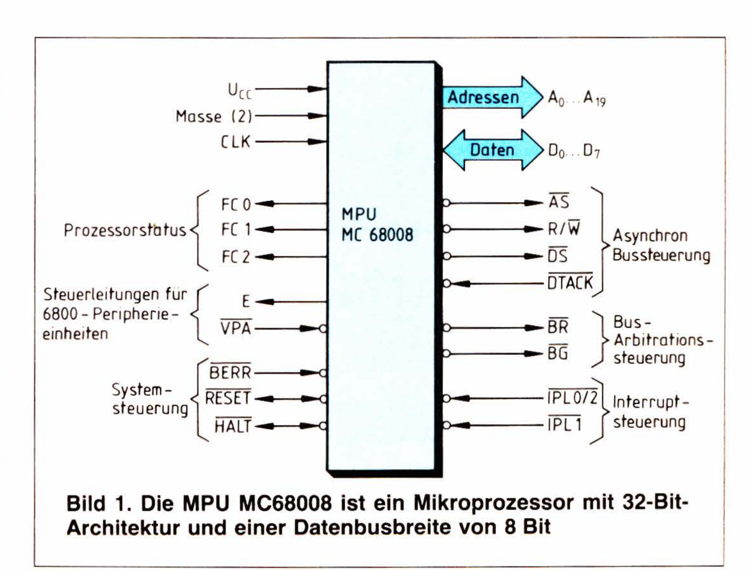


Bild 1. Die MPU MC68008 ist ein Mikroprozessor mit 32-Bit-Architektur und einer Datenbusbreite von 8 Bit

zugeschnitten. Bei einer langsamen Einheit fügt die MPU MC68008 Wartezyklen ein, bis DTACK empfangen wurde. In neuen Systementwürfen wird diese Technik dazu benutzt, den Systemdurchsatz zu maximieren und den Aufwand an externer Logik zu minimieren. In einem synchronen System (z. B. solche auf der Basis des 6809) sind Daten nach einem bestimmten Zeitintervall innerhalb des Buszyklus gültig. Bei einem MC6809-System ist dies bei der fallenden Flanke des E-Takt-Signals.

Es gibt zwei Methoden zum Anschluß des MC68008 an ein synchrones System: Die MPU selbst arbeitet synchron oder die MPU arbeitet asynchron.

MC68008 in synchroner Betriebsart

Der Baustein MC68008 arbeitet mit den Peripherie-Steuersignalen der M6800-Familie (E, VPA), die es der MPU ermöglichen, einen synchronen Buszyklus wie beim M6800 auszuführen. Um dies der MPU anzuzeigen, gibt die Peripherieeinheit anstelle von DTACK während des Buszyklus das VPA-Signal aus. Der E-Takt ist ein Zehntel des MC68008-Taktes (CLK), so daß, wenn die MPU z. B. einen 8-MHz-Takt verwendet, die Peripherieeinheit mit 800 kHz läuft. Das ergibt eine einfache Methode, synchrone Peripherieeinheiten mit minimalem Aufwand an externer Logik anzuschließen. Es gibt keinen Grund, warum der gesamte Speicher (einschließlich Programmspeicher) sowie Peripherieeinheiten nicht auf diese Weise gesteuert werden können, so daß jeder Buszyklus synchron ausgeführt wird. Allerdings bedeutet das, daß alle Buszyklen, einschließlich Instruction-Fetch- und Ausführungszyklen, eine große Zahl von Wartezyklen umfassen.

MC68008 in asynchroner Betriebsart

In diesem Fall können die Peripherieeinheiten vom M6800-Typ mit ihrer höchsten Betriebsfrequenz (2 MHz) getaktet werden, indem ein externer Generator den E-Takt erzeugt. Die MPU läuft asynchron, und das DTACK-Signal entsteht gleichzeitig mit dem „neuen" E-Takt. Damit wird die Anzahl der Wartezustände pro Buszyklus verringert. Bei diesem Aufbau sind mehr Schaltungselemente erforderlich, allerdings ergibt sich ein höherer Systemdurchsatz. Nach diesem Verfahren arbeitet die Schaltung, die im folgenden Abschnitt beschrieben ist.

3 Schaltung

Bild 3 zeigt die Blockschaltung der Kombination von MC68008 und M68MM19A. In *Bild 4* ist die Interface-Logik des MC68008 dargestellt. Alle Signale am Anschluß P1, Stift 1...40, sind über den DIL-Stecker des MC6809 mit der Platine M68MM19A verbunden. Der Datenbus des MC68008 und des M68MM19A werden über zwei 8fach-Latches SN74LS373 (U6 und U7) verbunden, die entgegengesetzt parallel angeschlossen sind. Deren Ausgangs-Enable-Anschlüsse werden über die R/W-Leitung der MPU gesteuert, so daß ein Latch (U7) beim MPU-Lese-Vorgang und das andere Latch (U6) beim MPU-Schreib-Vorgang freigegeben ist. Nor-

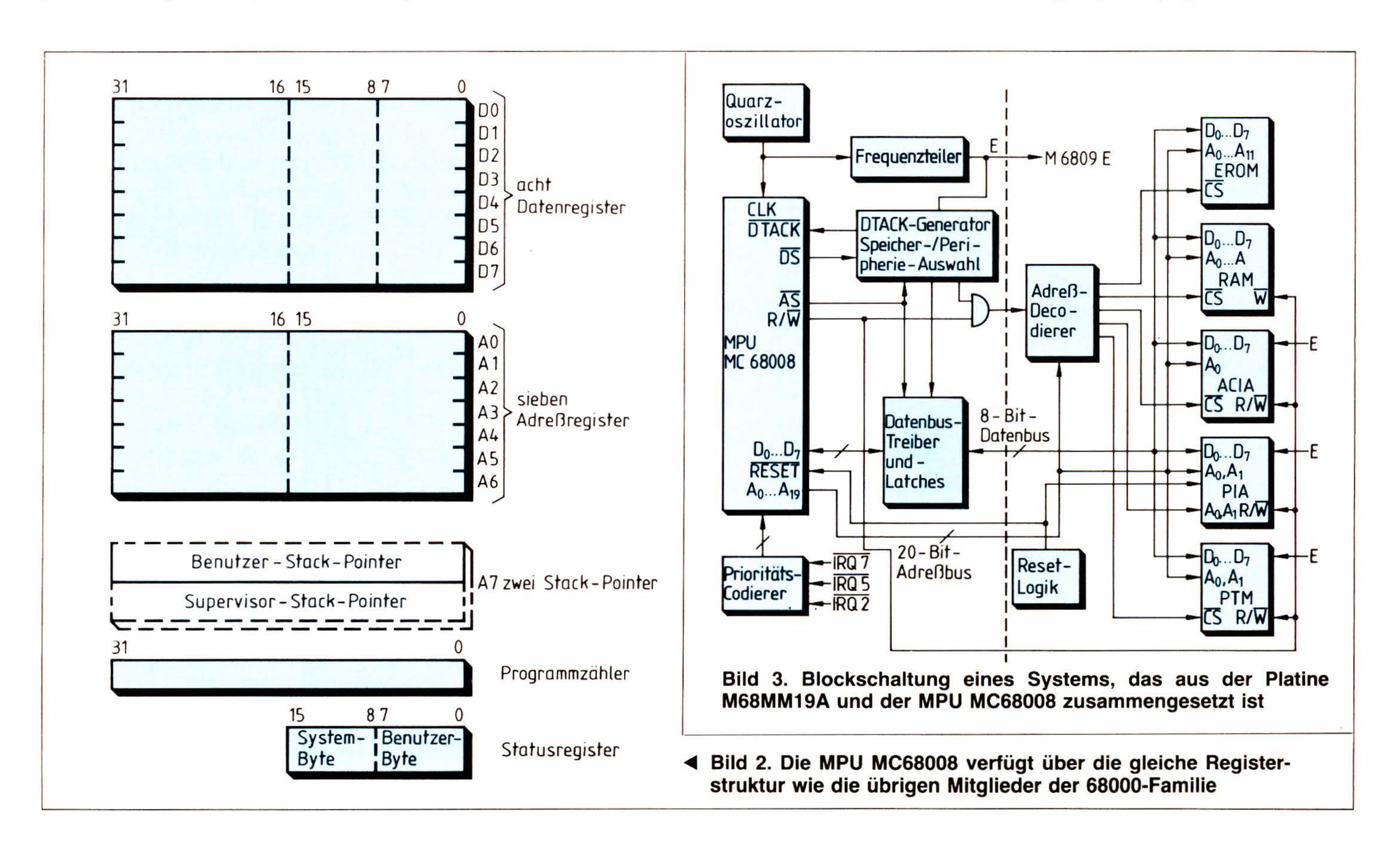


Bild 3. Blockschaltung eines Systems, das aus der Platine M68MM19A und der MPU MC68008 zusammengesetzt ist

◄ **Bild 2. Die MPU MC68008 verfügt über die gleiche Registerstruktur wie die übrigen Mitglieder der 68000-Familie**

malerweise befinden sich die Latches in einer transparenten Betriebsart, gehen aber in den Speicherbetrieb über, wenn die angeschlossenen Speicher- oder Peripherieeinheiten nicht angewählt sind. Der Speicherzustand hält an, bis das DTACK-Signal im Buszyklus-Zustand S7 des MC68008 umschaltet (die Buszyklus-Zustände sind im Datenblatt des MC68008 näher erläutert).

Mit Hilfe des JK-Flipflops SN74LS112 (U8A und U8B) wird das DTACK-Signal aus DS und E erzeugt. E wird aus dem Taktoszillator K1148 (U3) über den Zähler SN74LS163 (U4) abgeleitet. Das Signal am E-Ausgang des MC68008 wird nicht verwendet. Ein DTACK-Signal erhält die MPU in jedem Buszyklus, außer während eines Interrupt-Acknowledge-Zyklus (IACK). Während dieser Periode muß die unterbrechende Peripherieeinheit entweder DTACK (wenn die MPU eine Vektor-Interrupt-Sequenz ausführen muß) ausgeben, wobei die Peripherieeinheit die Nummer des Interruptvektors auf den Datenbus geben muß, oder sie muß VPA (für die MPU, die eine Auto-Vektor-Sequenz ausführt) ausgeben. Weil keine der Peripherieeinheiten auf der Platine M68MM19A in der Lage ist, eigene Interrupt-Vektor-Nummern auszugeben, muß DTACK während des IACK-Zyklus unterdrückt und VPA an dessen Stelle eingesetzt werden. Den IACK-Zustand zeigen die Funktionscodes des MC68008 an (FC1 = FC2 = FC3 = 1). Zur Erzeugung des VPA und zum Sperren des DTACK sind

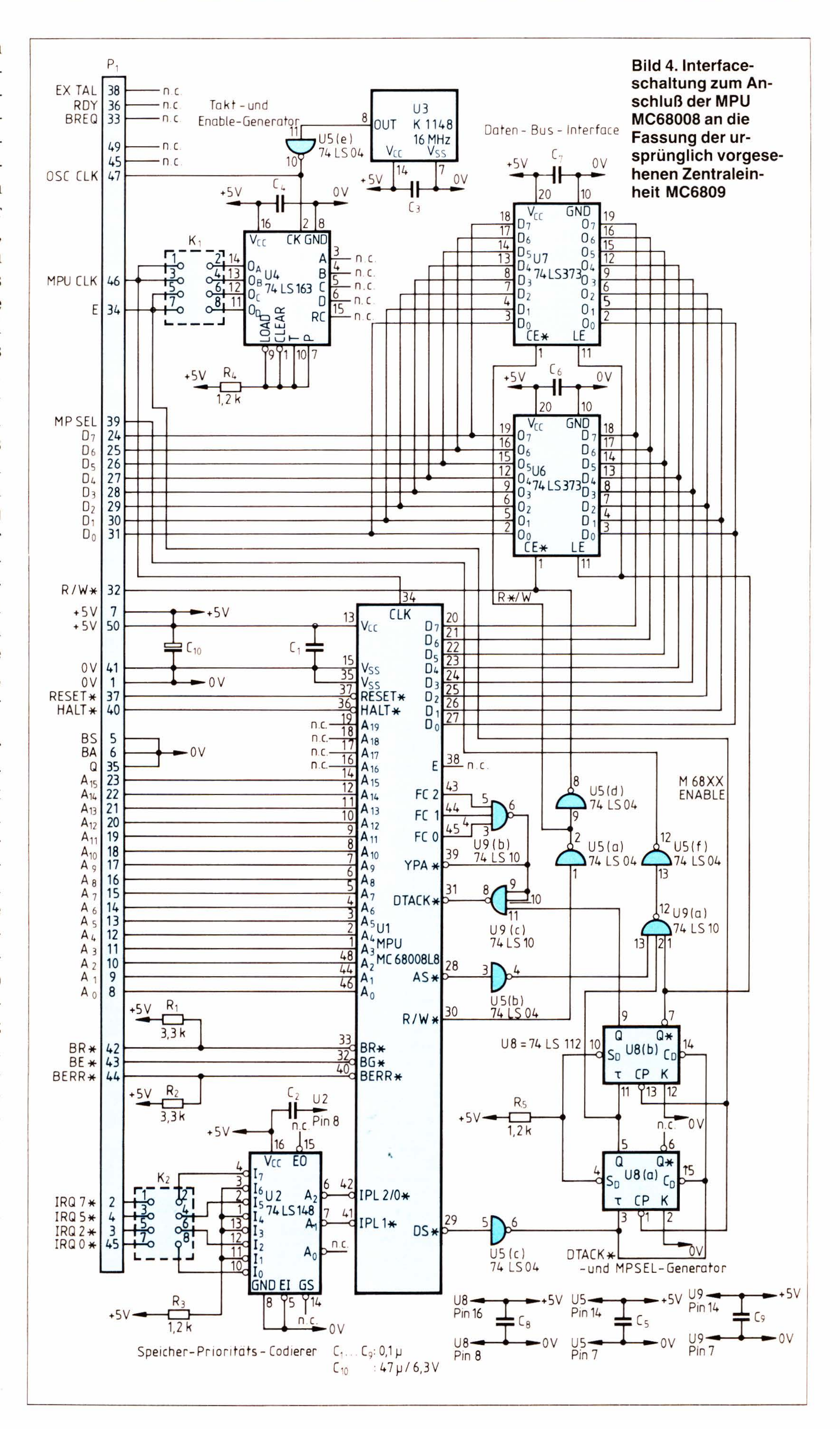


Bild 4. Interfaceschaltung zum Anschluß der MPU MC68008 an die Fassung der ursprünglich vorgesehenen Zentraleinheit MC6809

die NAND-Gatter U9B und U9C (SN74LS10) vorgesehen.

Das Signal „MPSEL“ aus dem Inverter SN74LS04 (U5F) wird verwendet, um die Adreß-Decodierungslogik der Platine M68MM19A für Speicher- und Peripherieeinheiten freizugeben. *Bild 5* zeigt die zeitlichen Verhältnisse zwischen den MPUs MC6809 und MC68008 bei einer Lese-Operation. MPSEL wird zwischen den Zuständen S3, der den Beginn des MC6809-Zyklus anzeigt (A0...A19, AS, R/W und DS werden in den Zuständen S0...S2 vorbereitet). Die Daten vom Speicher oder aus Peripherieeinheiten werden vor der nächsten fallenden Flanke von E gültig. Zu diesem Zeitpunkt übernimmt U7 die Daten in den Zwischenspeicher und DTACK wird an die MPU angelegt. Während S6 liest die MPU MC68008 Daten aus dem Latch U7, und der Buszyklus fließt mit dem Ende von S7 ab.

In ähnlicher Weise werden während einer MPU-Schreiboperation die Daten aus der MPU bei S2 und DS bei S3 gültig. Daten werden vom Speicher oder der Peripherieeinheit bei der nächsten fallenden Flanke von E zwischengespeichert. DTACK wird ausgegeben, und die MPU schließt den Buszyklus ab.

Zähler U4 und die Drahtbrücken-Programmierung K1 ermöglichen der MPU sowie den Speicher- und Peripherieeinheiten mit verschiedenen Taktfrequenzen zu arbeiten, z. B. 4 oder 8 MHz für die MPU MC68008 und 1 oder 2 MHz für Speicher und Peripherie. Der Prioritätscodierer SN74LS148 (U2) sowie die Drahtbrückenprogrammierung K2 werden zur Festlegung der relativen Priorität des Peripherie-Interrupts benutzt. Es ist zu beachten, daß im Gegensatz zum MC68008, der über drei Interrupt-Prioritätsebenen-Eingänge verfügt (IPL0, IPL1 und IPL2), die 48polige Version des MC68008 lediglich zwei davon besitzt (IPL0/2 und IPL1). IPL0 und IPL2 sind intern miteinander verbunden. Daraus ergibt sich, daß lediglich vier der acht Interruptebenen (0, 2, 5 und 7) benutzbar sind.

4 Praktische Erfahrungen

Verbindungen zwischen der MC68008-Platine und dem Computer M68MM19A erfolgen über den Stecker P1 (50poliger Anpreß-Stecker) auf der MC68008-Platine und über die DIL-Fassung des MC6809 auf der Computerplatine. Wie aus Bild 4 hervorgeht, sind die Signale auf dem Stecker P1 (Anschlüsse 1...40) der direkte Ersatz für diejenigen des MC6809 (außer Anschluß 39 – MPSEL-Signal, das zur Freigabe des Adressen-Decodierers benutzt wird).

Durch Neuprogrammierung des FPLA-Adreßdecodierers auf der Computerplatine M68MM19A werden die abweichenden Anforderungen des MC68008 angepaßt (z. B. liegt der Reset und andere Vektoren beim MC68008 in Speicherbereichen geringerer Ordnung).

Mehrere Benchmark-Tests wurden durchgeführt, bei denen die unveränderte 8-Bit-Platine M68MM19A mit der 2-MHz-Version der MPU MC6809 mit dem MC68008/M68MM19A-System verglichen wurde. Es zeigte sich eine drei bis vierfache Erhöhung der Arbeitsgeschwindigkeit gegenüber der Ursprungsausführung. Allerdings kann man dabei nicht direkt auf die Leistung des MC68008 schließen, weil die MPU im Betrieb Wartezyklen einfügen muß, bis die Platine M68MM19A in der Lage ist, Daten zu transferieren. In einem neu entwickelten System, bei dem MPU und Speicher- bzw. Peripherieeinheiten asynchron betrieben werden, ist es möglich, ohne Wartezyklen im Buszyklus zu arbeiten und damit eine maximale Datentransferrate zu erreichen (Faktor 4...5). Derzeit sind vom MC68008 Versionen für eine Taktfrequenz von 10 und 12,5 MHz erhältlich, eine 16-MHz-Ausführung ist geplant.

Literatur

[1] von Bechen, P.: 8-Bit-Prozessor bietet 32-Bit-Architektur. ELEKTRONIK 1983, H. 1, S. 43...46.

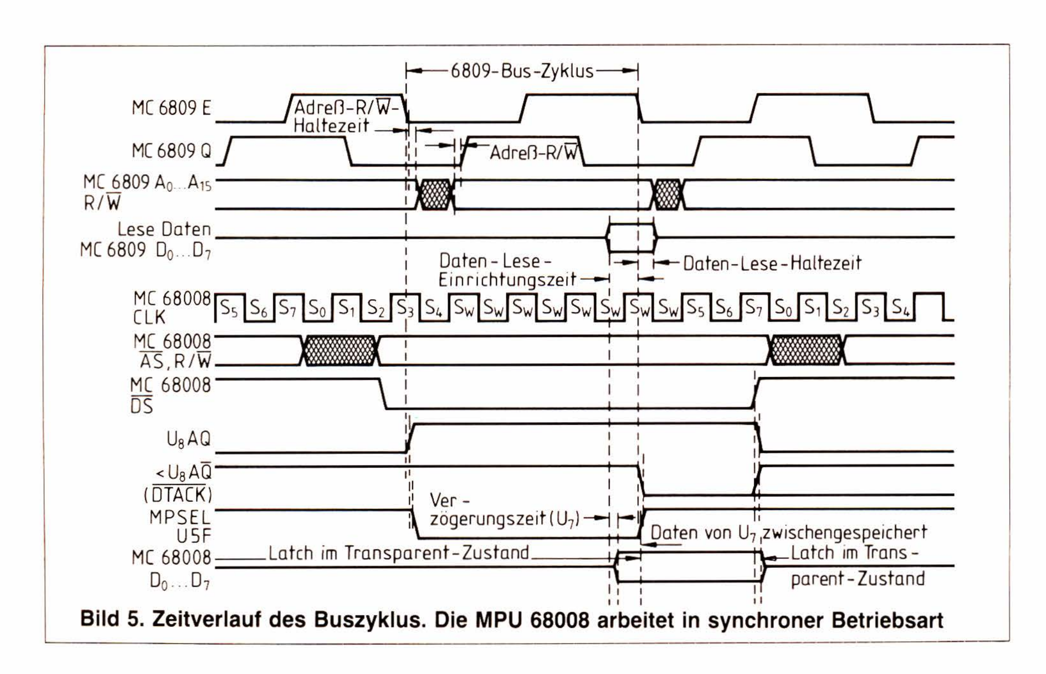


Bild 5. Zeitverlauf des Buszyklus. Die MPU 68008 arbeitet in synchroner Betriebsart


Dipl.-Physiker Andy Barth ist Engländer. Seit 1976 ist er tätig in den Bereichen Mikroprozessor-System-Design und -Applikationen in Deutschland, USA und Großbritannien. Zur Zeit arbeitet er als Senior Systems Engineer bei Motorola in East Kilbride, Schottland.

Dipl.-Ing. Jürgen Hülsemann, Dr.-Ing. Zoltán Benyó

Ausnahmebehandlung der Mikroprozessoren MC68000 und MC68010

Die unterschiedliche Behandlung der Ausnahmen der 68000-Prozessoren wurde bei Arbeiten zur Erstellung eines Selbsttestprogramms für diese Prozessoren entdeckt. Beim Test der Privilegverletzung, d. h. bei der Ausführung von privilegierten Befehlen im Benutzerzustand, wurde mit dem eigenen Ausnahmebehandler eine Endlosschleife ausgeführt. Dabei passierte folgendes: Der Ausführungsversuch des privilegierten Befehls im Benutzerzustand erzeugt die Privilegverletzungsausnahme. Bei der Ausnahmebehandlung wird das Statusregister und als Rücksprungadresse die Adresse des privilegierten Befehls auf dem Systemstapel abgelegt. Am Ende der Ausnahmebehandlung wird mit dem Befehl RTE (Return-from-Exception) das Statusregister und die Rücksprungadresse zurückgeladen. Dadurch wird anschließend der privilegierte Befehl im Benutzerzustand wiederholt usw. (Endlosschleife). Bei der Durchsicht der Literatur über den MC68000 fanden sich keine genauen Hinweise über die Behandlung der verschiedenen Ausnahmen. Dies hat die Verfasser veranlaßt, deren Bearbeitung und Behandlung zu analysieren.

System- und Benutzermodus

Der Prozessor unterscheidet zwischen zwei Betriebsmodi, dem System- und dem Benutzermodus. Während im Systemmodus alle Befehle ausgeführt werden können, ist im Benutzermodus die Ausführung der privilegierten Befehle verboten. Diese betreffen das Laden und Verändern des Statusregisters (MOVE to SR, ANDI to SR,...). Der Zugriff auf den Benutzerstapelzeiger im Systemmodus (MOVE to/from USP) ist auch privilegiert. (Im Benutzermodus kann prinzipiell nicht auf den Systemstapelzeiger zugegriffen werden.) Weitere privilegierte Befehle sind der RESET-, RTE- und der STOP-Befehl. Außerdem kann auf das Systembyte des Statusregisters im Benutzermodus nur lesend zugegriffen werden.

Der Betriebsmodus wird im Systembyte des Statusregisters vom Überwachungsbit (Bit 13) angezeigt. Eine logische Eins steht dabei für den Systemmodus und eine logische Null für den Benutzermodus. Die Übergänge vom Benutzermodus in den Systemmodus und umgekehrt sind in *Bild 1* dargestellt.

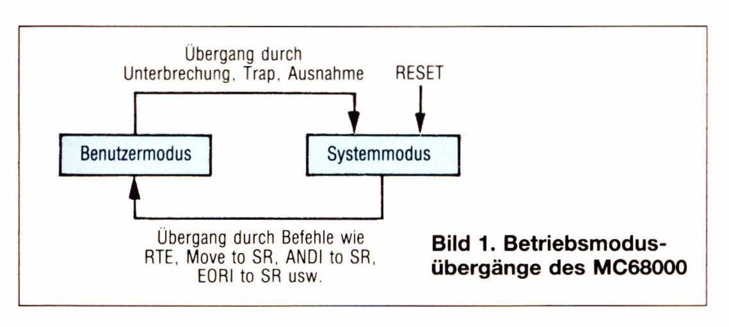


Bild 1. Betriebsmodusübergänge des MC68000

Ausnahmebehandlung

Der MC68000 hat drei Zustände:

1. Den Normalzustand, d. h. das Programm läuft normal ab (im System- oder Benutzermodus).
2. Den Ausnahmezustand; das ist der Zustand, in dem die Ausnahmen behandelt werden (stets im Systemmodus).
3. Den Halt-Zustand; dieser wird erreicht, wenn von außen die Halt-Leitung aktiviert ist, oder intern bestimmte Fehler erkannt sind, z. B. der doppelte Busfehler.

Es gibt zwei Gruppen von Ausnahmen, nämlich intern und extern erzeugte; interne werden bei der Befehlsausführung erzeugt, während externe von der Hardware an den entsprechenden Anschlüssen des Prozessors ausgelöst werden. Einen Überblick gibt die *Tabelle 1*.

Jeder Ausnahme ist ein bestimmter Ausnahmevektor zugeordnet. Diese Vektoren sind Nummern, aus denen durch Multiplikation mit vier Adressen gewonnen werden; deren Inhalt stellt erst die Adresse für die Ausnahmebehandlungsroutine zur Verfügung. Dabei sind für

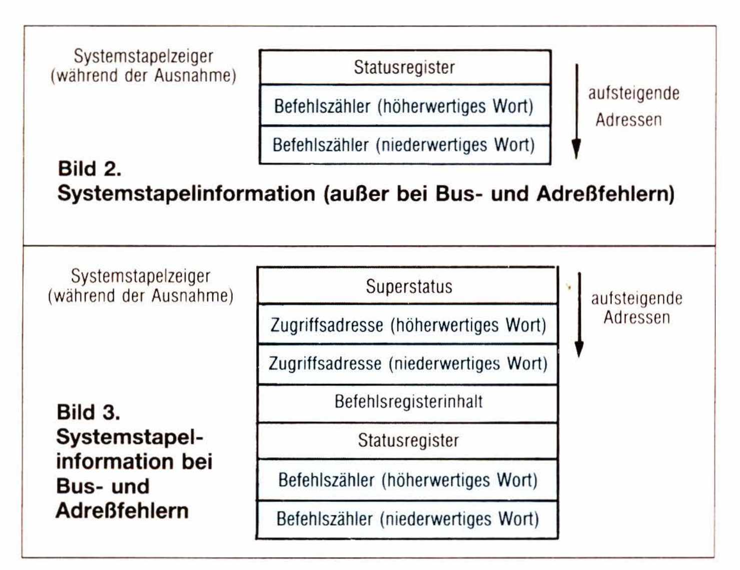


Bild 2. Systemstapelinformation (außer bei Bus- und Adreßfehlern)

Bild 3. Systemstapelinformation bei Bus- und Adreßfehlern

die einzelnen Adressen jeweils 4 Byte reserviert. Die Ausnahmevektoren liegen dabei hintereinander im Speicher beginnend bei der Speicheradresse Null.

Die Ausnahmebehandlung findet prinzipiell im Systemmodus statt. Zusätzlich wird das Einzelschritt-Bit zurückgesetzt. Zu Beginn der Ausnahmebehandlung werden die folgenden Informationen auf dem Systemstapel abgelegt (*Bild 2*), um das Programm an der unterbrochenen Stelle nach der Ausnahmebehandlung wieder fortsetzen zu können.

Bei Bus- und Adreß-Fehlern legt der Prozessor aus Diagnosegründen zusätzliche Informationen auf dem Systemstapel ab (*Bild 3*).

Abgeschlossen werden die einzelnen Ausnahmebehandlungsroutinen mit dem Befehl RTE (Return-from-Exception). Bei der Ausführung dieses Befehls werden das Statusregister und der Befehlszähler wieder vom Systemstapel zurückgeladen, anschließend wird das unterbrochene Programm weiterbearbeitet.

Die einzelnen Ausnahmen

Bei herkömmlicher Ausnahmebehandlung zeigt der auf dem Systemstapel abgelegte Befehlszähler auf den bei normalen Programmablauf nächsten ausführbaren Befehl. Dies ist bei den Prozessoren MC68000 aber nicht immer so. Diese Sonderfälle sind bei den im folgenden beschriebenen Ausnahmen jeweils gesondert vermerkt. Zusätzlich ist bei den einzelnen Ausnahmen jeweils ihre Vektornummer VN angegeben.

1. RESET — *VN: 0,1*

Beim Rücksetzen der Hardware wird der Inhalt des ersten Doppelwortes aus der Vektortabelle in den Systemstapelzeiger und der des zweiten Doppelwortes in den Befehlszähler geladen. (Dies ist der einzige Fall, bei dem auch der Systemstapelzeiger geladen wird; denn hier wird keine Information auf dem Systemstapel abgelegt.)

2. Busfehler — *VN: 2*

Der Busfehler wird über einen externen Anschluß des Prozessors an die Ablaufsteuerung gemeldet. Mögliche Busfehlerquellen sind: Zeitüberwachung der Buszyklen, Speicherfehler, Fehlermeldungen einer Speicherverwaltungseinheit, usw. Die Ausnahmebehandlung beginnt am Ende des laufenden Taktzyklus. Dabei werden zusätzliche Informationen für eine mögliche Diagnose auf dem Systemstapel abgelegt (*Bild 3*). Der hierbei abgelegte Befehlszähler kann jedoch aufgrund des vorausschauenden Befehlsholens („Instruction Prefetch") von der aktuellen Befehlsadresse abweichen, da er gewöhnlich auf das letzte eingelesene Wort der Befehlssequenz zeigt. Dabei ist es möglich, daß auch das Befehlsregister schon den nächsten Befehlscode beinhaltet. Aus diesen und weiteren Gründen erlaubt der MC68000 bei einer Bus-Fehler-Ausnahme prinzipiell keine Fehlerbehebung.

3. Adreßfehler — *VN: 3*

Dieser Fehler tritt bei Wort- oder Doppelwortzugriffen auf ungerade Adressen auf. Dabei gilt für die Adreßfehler-Ausnahme dasselbe wie für die Busfehler-Ausnahme, nämlich daß keine Fehlerbehebung möglich ist. Der einzige Unterschied ist, daß diese Ausnahme intern erzeugt wird.

4. Illegaler Befehl — *VN: 4*

Wenn im Programm eine nicht implementierte Bitkombination als Befehlscode auftritt, wird diese Ausnahme ausgelöst. Dabei werden die Befehlscodes 4AFA und 4AFB von Motorola in deren Entwicklungssystemen verwendet, der Befehlscode 4AFC steht dem Benutzer zur Verfügung. Diese drei Befehlscodes werden auch bei den Nachfolgetypen des Prozessors illegal bleiben. Beim Ablegen der Information auf dem Systemstapel wird die Adresse des illegalen Befehls (nicht die des Folgebefehls) abgelegt.

5. Division durch Null — *VN: 5*

Diese Ausnahme tritt auf, wenn der Nenner bei einer Division Null ist.

6. Befehl CHK — *VN: 6*

Befindet sich der Wert des im Befehl spezifizierten Datenregisters außerhalb der im Befehl angegebenen Grenzen, so erfolgt diese Ausnahmebehandlung.

7. Befehl TRAPV — *VN: 7*

Hier handelt es sich um einen bedingten TRAP-Befehl; er wird ausgeführt, wenn das Überlaufbit V im Statusregister (Bit 1) während der Befehlsausführung gesetzt ist.

8. Privilegverletzung VN: 8

Die Privilegverletzungsausnahme erfolgt, wenn versucht wird, im Benutzermodus einen privilegierten Befehl auszuführen. Hier wird auf dem Systemstapel die Adresse des aufgerufenen (privilegierten) Befehls abgelegt.

9. Einzelschritt VN: 9

Ist das Einzelschritt-Bit T im Statusregister (Bit 15) gesetzt, so wird nach jedem Befehl in das mit diesem Vektor definierte Programm gesprungen.

10. Nicht implementierte Befehle VN: A,B

Ein Befehl mit dem Befehlscode Axxx oder Fxxx erzeugt diese beiden Ausnahmebehandlungen. Auf dem Systemstapel wird jeweils die Adresse des aufgerufenen (nicht implementierten) Befehls abgelegt.

11. Illegales Format VN: E

Diese Ausnahme wird im MC68000 nicht verwendet. (Siehe Abschnitt „Unterschiede zwischen MC68000 und MC68010".)

12. Nicht initialisierte Unterbrechung VN: F

Dieser Vektor wird von nicht initialisierten MC68000-Peripheriebausteinen ausgegeben, falls sie nicht in der Auto-Vektor-Betriebsart arbeiten.

13. Falsche Unterbrechung VN: 18

Wenn während der Unterbrechungsbestätigung ein Busfehler auftritt, wird diese Ausnahmebehandlung durchgeführt.

14. Autovektor-Unterbrechung VN: 19-1F

Diese Ausnahmebehandlungen werden bei externen Unterbrechungen ausgeführt, wenn das Signal $\overline{VPA}$ gesetzt ist. Dann wird der entsprechende Unterbrechungsvektor intern aufgrund der anliegenden Unterbrechungspriorität ausgewählt. Die externe Unterbrechungspriorität muß dabei höher sein als die im Statusregister abgespeicherte. Andernfalls erfolgt keine Unterbrechung.

15. TRAP-Befehle VN: 20-2F

Bei der Ausführung der Befehle TRAPi wird automatisch die entsprechende Ausnahmebehandlung durchgeführt (i = 0...15).

16. Benutzerdefinierte Unterbrechungsvektoren VN: 40-FF

Diese Unterbrechungsvektoren werden extern generiert und als Vektornummer über den Datenbus eingelesen. Auch hier muß die Unterbrechungspriorität höher sein als die im Statusregister abgespeicherte.

Tabelle 1. Die verschiedenen Ausnahmen des MC68000

interne Ausnahmen	Befehle (TRAP, TRAPV, CHK, DIV) Adreßfehler beim Datenzugriff Einzelschritt-Modus Illegale und nicht implementierte Befehle Privilegverletzung
externe Ausnahmen	Unterbrechungen Bus-Fehler RESET

Die fehlenden Ausnahmevektoren sind von Motorola noch nicht spezifiziert und für spätere Anwendungen reserviert. In *Tabelle 2* werden die unterschiedlichen Ausnahmen nach der Priorität, dem Zeitpunkt des Beginns der Ausnahmebearbeitung und dem auf dem Systemstapel abgespeicherten Befehlszähler zusammengestellt. Die einzelnen Ausnahmen sind geordnet nach ihrer Priorität in Gruppen zusammengefaßt; dabei hat die Gruppe 0 die höchste und die Gruppe 2 die niedrigste Priorität. Auch innerhalb einer Gruppe ist die Priorität nach oben hin aufsteigend. Eine laufende Ausnahmebehandlung kann jeweils nur von einer Ausnahme mit höherer Priorität unterbrochen werden.

Konsequenzen für die Ausnahmebearbeitung

Bei Bus- und Adreßfehler-Ausnahmen kann der MC68000 nur eine Fehlermeldung ausgeben; eine Fehlerbehandlung ist jedoch nicht möglich. Dies ist dadurch bedingt, daß die auf dem Systemstapel abgelegten Informationen ungenügend sind und keine Befehlswiederholung oder Weiterverarbeitung des abgebrochenen Befehls erlauben.

Bei den illegalen und nicht implementierten Befehlen sowie bei der Privilegverletzung ist eine Ausnahmebehandlung und anschließende Fehlerbehebung technisch möglich. Hier wird jeweils die Adresse des abgebrochenen Befehls als Rücksprungadresse auf dem Systemstapel abgelegt. Bei diesen Ausnahmen muß vor der Fortsetzung des Programms an der unterbrochenen Stelle die Rücksprungadresse modifiziert werden. Andernfalls

Tabelle 2. Übersicht über die verschiedenen Ausnahmen

Gruppe	Ausnahme	Beginn der Ausnahmebearbeitung	Abgespeicherter Befehlszähler
0	RESET Busfehler Adreßfehler	am Ende eines Taktzyklus	– durch Prefetch beeinflußt
1	Einzelschritt Unterbrechung	am Ende eines Befehlszyklusses	nächster Befehl
	Illegaler Befehl Nicht impl. Befehl Privilegverletzung	am Ende eines Buszyklusses	dieser Befehl
2	TRAP, TRAPV, CHK Division durch Null	innerhalb dieses Befehlszyklusses	nächster Befehl

würde die eingangs beschriebene Endlosschleife entstehen. Das Ablegen der Adresse des abgebrochenen Befehls auf dem Systemstapel ist sinnvoll, weil dem Prozessor bei den illegalen und den nicht implementierten Befehlen die Befehlslänge nicht bekannt ist und er deshalb nicht die Adresse des nächsten Befehls bestimmen kann. Die Verwendung dieser Befehle ist durch den Benutzer zur Realisierung von eigenen Befehlen möglich, da er die entsprechenden Befehlscodes in sein Programm einbauen und somit gezielt verwenden kann. Bei der Privilegverletzung muß in der Ausnahmebehandlung auf die Fehlerursache reagiert werden; auch dazu ist der unberechtigt ausgeführte privilegierte Befehl notwendig. Eine Fehlerbehandlung ist bei der Privilegverletzung sehr komplex und nicht immer möglich, weil die Fehlerursache schwer identifizierbar ist.

Bei allen anderen Ausnahmen zeigt der auf den Systemstapel abgelegte Befehlszähler auf den nächsten Befehl, so daß im Normalfall die Rücksprungadresse während der Unterbrechungsbearbeitung nicht modifiziert werden muß. Hier ist die Programmfortsetzung jeweils mit dem nächsten Befehl vorgesehen – die Ausnahmen TRAPV, CHK und Division durch Null sind bedingte Ausnahmen, die eine anforderungsabhängige Behandlung durch den Benutzer erfordern. Wenn keine Fehlerbehandlung möglich ist, muß das jeweilige Programm abgebrochen werden. Man muß hier klar zwischen der technischen Möglichkeit der Programmfortsetzung und ihrer sinnvollen Anwendung unterscheiden.

Unterschiede zwischen MC68000 und MC68010

Der wesentliche Unterschied zwischen dem MC68000 und dem MC68010 ist der, daß der Prozessor MC68010 auch nach einer Bus- oder Adreßfehler-Ausnahme zu seinem Programm zurückkehren kann. Das ist dadurch möglich, daß der MC68010 bei diesen beiden Ausnahmen zusätzlich zu den vom MC68000 abgespeicherten Informationen auch prozessorinterne Zustände auf dem Systemstapel ablegt, die es erlauben, einen einmal begonnenen Befehl an der abgebrochenen Stelle durch Wiederholung des fehlerhaften Buszyklus fortzusetzen. Dabei kann in der Fehlerbehandlung entschieden werden, ob der fehlerhafte Zugriff durch den Prozessor wiederholt oder durch eine Behandlungsroutine ersetzt werden soll. Die Busfehler-Ausnahme wird zur Implementierung eines virtuellen Speichers benötigt, da dann bei einem Seitenfehler die fehlende Seite nachgeladen und anschließend der unterbrochene Befehl fortgesetzt werden kann. Dies ist ein Beispiel für einen Hardware-Fehler, bei dem die Fehlerkorrektur möglich und eine damit verbundene Programmfortsetzung sinnvoll ist. Für weitere Hardware-Fehler muß die Fehlerbehandlungsroutine feststellen, ob der Fehler korrigierbar ist. Falls ja, kann das Programm fortgesetzt werden, sonst muß das Programm abgebrochen werden.

Der MC68010 legt bei einer Ausnahme folgende Informationen auf dem Systemstapel ab (*Bild 4*):

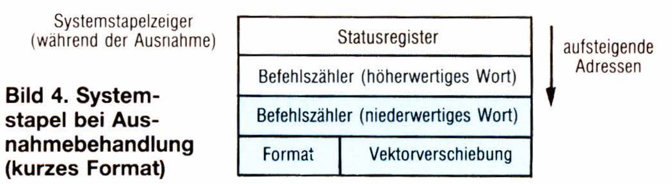


Bild 4. Systemstapel bei Ausnahmebehandlung (kurzes Format)

Dazugekommen ist hier das letzte Wort, welches das Format und die Vektorverschiebung (vector offset) enthält. Die höchstwertigen 4 Bit beinhalten dabei das Format; 0000 bedeutet kurzes Format (verwendet bei allen Ausnahmen außer Bus- und Adreßfehler) und 1000 langes Format. Daran erkennt der Prozessor MC68010 bei der Ausführung des Befehls RTE (Return from Exception), wie viele Informationen vom Systemstapel zur Programmfortsetzung zurückgeladen werden müssen. Die niederwertigen 10 Bits geben die Vektorverschiebung der aktuellen Ausnahme an; mit dieser wurde durch Addition zu dem Vektor-Basis-Register die Adresse gebildet, die die Startadresse der Ausnahmebehandlungsroutine beinhaltet. Mit dem Vektor-Basis-Register ist es möglich, die Vektortabelle zu verschieben bzw. im Betrieb auf eine andere umzuschalten.

Bei der Bus- und Adreßfehler-Ausnahme werden zusätzlich auch die für eine Weiterbearbeitung des abgebrochenen Befehls notwendigen Informationen auf dem Systemstapel abgelegt (siehe Bild 5). Achtung, die Reihenfolge von Befehlszähler und Zugriffsadresse wurde gegenüber dem MC68000 (vergleiche auch Bild 3) geändert! Auch das Superstatuswort wurde erweitert.

Weiterhin kann im Benutzermodus nicht mehr lesend auf den Systemteil des Statuswortes zugegriffen werden.

Als weitere Ausnahme hat der MC68010 das illegale Format (VN: E). Diese Ausnahme wird ausgelöst, wenn der Prozessor bei einem Befehl RTE (Return from Exception) ein ungültiges Format von seinem Systemstapel liest, oder wenn das erste der sechzehn Worte mit internen Informationen über den Prozessorzustand, welches auch die Prozessorversionsnummer enthält, ungültig ist.

Systemstapelzeiger (während der Ausnahme) → aufsteigende Adressen ↓

Statusregister	
Befehlszähler (höherwertiges Wort)	
Befehlszähler (niederwertiges Wort)	
Format	Vektorverschiebung
Superstatus	
Zugriffsadresse (höherwertiges Wort)	
Zugriffsadresse (niederwertiges Wort)	
UNBENUTZT, RESERVIERT	
Daten-Ausgabe-Puffer	
UNBENUTZT, RESERVIERT	
Daten-Eingabe-Puffer	
UNBENUTZT, RESERVIERT	
Befehls-Eingabe-Puffer	
Interne Informationen über den Prozessorzustand; 16 Worte	

Bild 5. Systemstapel bei Bus- und Adreßfehler-Ausnahmen (langes Format)

Dipl.-Phys. Hans-Jürgen Nischik

68008 ersetzt Z80

Im Institut für angewandte Physik der Universität Münster wird seit einiger Zeit an der Entwicklung eines 16-/32-Bit-Multi-Mikroprozessor-/Multi-Bus-Systems gearbeitet. Zur Zeit finden die CPU und die Peripherie-Bausteine der 68000-Familie Verwendung. Um für ein solches System erste Betriebssoftware erstellen zu können, benötigt man einen lauffähigen Entwicklungsrechner, der mit der gleichen CPU arbeitet wie der zu entwickelnde Rechner. Die Kosten dafür sind allerdings noch sehr hoch. Deshalb wurde nach einer Möglichkeit gesucht, ein Entwicklungssystem aufzubauen, das mit relativ geringem finanziellem Aufwand realisierbar ist. So entstand eine CPU-Karte im Einfach-Europaformat, die mit der MPU 68008 bestückt ist und die auf dem zum ECB-Bus aufwärtskompatiblen „VAMOS-80"-Bus betrieben werden kann.

Der Anwender kann nun auf eine Vielzahl von kostengünstigen Speicher- und Peripherie-Karten zurückgreifen, um sich ein System für 68000-Anwendungen aufzubauen. Das Zeitverhalten der 68008-Karte entspricht dem Timing einer Z80-CPU.

Die höhere Taktfrequenz, die Möglichkeit, 1 MByte Speicher zu adressieren und der sehr viel effektivere Befehlssatz der CPU 68008 ermöglichen eine Leistungssteigerung gegenüber einer Z80-CPU von bis zu 500 %. In einigen Fällen, zum Beispiel bei 32-Bit-Multiplikationen und -Divisionen, beträgt die Leistungssteigerung mehr als das Hundertfache.

Mikroprozessor 68008

Der Mikroprozessor 68008 verfügt über einen 8 Bit breiten Datenbus sowie einen 20 Bit breiten Adreßbus. Er ist vollständig softwarekompatibel zum 68000. Die Möglichkeiten des prioritätsgestaffelten Vektor-Interrupts sind von sieben Ebenen beim 68000 auf drei Ebenen beim 68008 reduziert.

„VAMOS-80"-Bus als Backplane

Viele Z80-Anwender, die mit ECB- oder VAMOS-80-Bussystemen arbeiten, kennen das Problem, mit lediglich 64 KByte linearem Adreßraum auszukommen. Auch Banking-Verfahren, die komplizierte Softwaremodule erfordern, können dieser Problematik nur schwer gerecht werden. Der Mikroprozessor 68008 löst diese Schwierigkeit auf Grund seines 1 MByte umfassenden

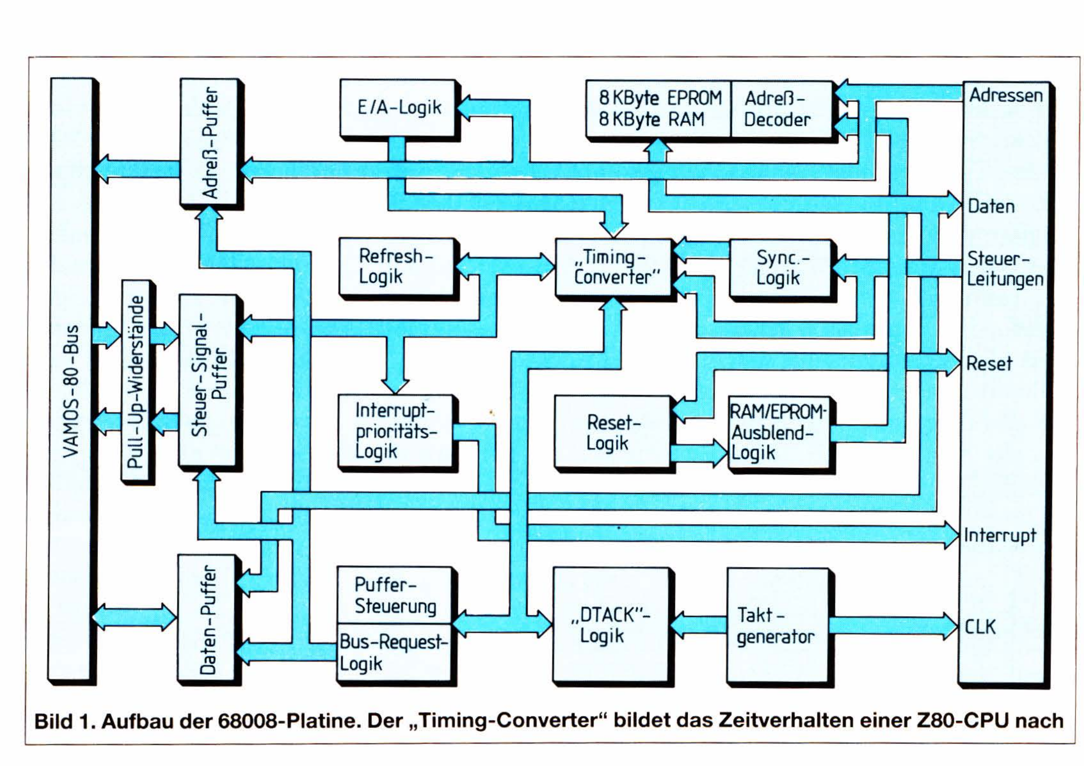


Bild 1. Aufbau der 68008-Platine. Der „Timing-Converter" bildet das Zeitverhalten einer Z80-CPU nach

linearen Adreßraums. Da der VAMOS-80-Bus über 20 Adreßleitungen verfügt, mit denen man in der Lage ist, 1 MByte Adressen zu erzeugen, eignet er sich sehr gut dazu, die Möglichkeiten der CPU 68008 auszuschöpfen.

Um den prioritätsgestaffelten Vektor-Interrupt auf drei Ebenen voll nutzen zu können, wurde der VAMOS-80-Bus um ein drittes Interrupt-Signal erweitert. Es stehen nunmehr die Interrupt-Signale INT1=INT, INT2 und NMI zur Verfügung.

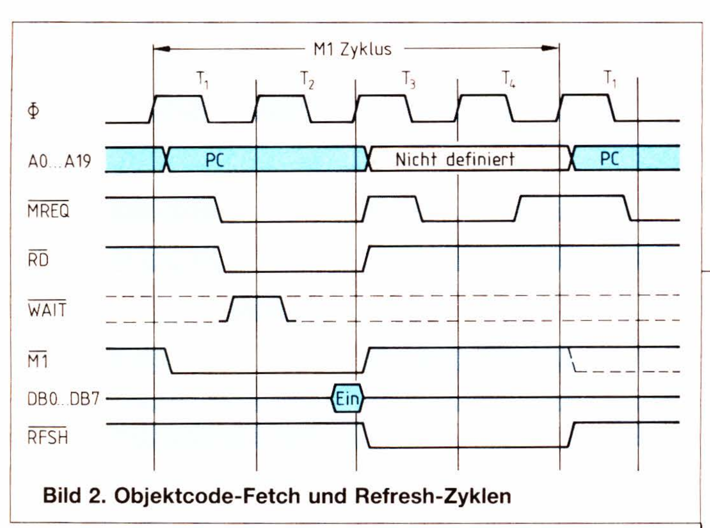


Bild 2. Objektcode-Fetch und Refresh-Zyklen

Damit eine 68008-CPU-Karte gegen eine Z80-CPU-Karte austauschbar wird, muß gewährleistet sein, daß sich das Zeitverhalten beider Karten entspricht. Zu diesem Zweck und aus Gründen, die in den folgenden Abschnitten deutlich werden, enthält die 68008-CPU-Karte insgesamt 12 logische Funktionsgruppen (*Bild 1*).

Die Aufgabe des „Timing-Converters" besteht darin, die Steuersignale, die von der 68008 generiert werden, in Z80-Steuersignale zu übersetzen.

Der Taktgenerator liefert sowohl den Takt für die 68008-CPU als auch den Takt für den VAMOS-80-Bus. Man muß jedoch berücksichtigen, daß die langsamste 68008-Version mit 8 MHz arbeitet. Mit dieser hohen Taktfrequenz sind die meisten Peripheriebausteine überfordert. Deshalb wird der durch zwei geteilte 68008-Takt als Systemtakt auf den Bus geführt.

Die Aufgabe der Synchronisationslogik besteht darin, Adreß-Strobe und Daten-Strobe, die von der CPU 68008 erzeugt werden, mit dem Systemtakt zu synchronisieren.

Die CPU benötigt beim Zugriff auf Speicher oder Peripheriebausteine ein asynchrones Handshake-Signal. Das heißt, daß sie bei jedem Zugriff das Quittungssignal „DTACK" erwartet. Da dieses Signal nicht auf ECB- oder VAMOS-80-Bus-kompatiblen Europa-Karten erzeugt wird, muß es die CPU-Karte generieren. Das DTACK-Signal kann bei Speicherzugriffen und bei Zugriffen auf Peripheriebausteine bis zu vier Systemtaktperioden verzögert werden. Außerdem berücksichtigt die DTACK-Logik das WAIT-Signal, das von langsamer externer Peripherie zu Synchronisationszwecken erzeugt wird.

Bei einem Objectcode-Fetch benötigt eine CPU Z80 zwei Taktperioden, während sie beim Schreiben oder Lesen von Daten drei Taktperioden beschäftigt ist. Die DTACK-Logik ist so ausgelegt, daß auch beim Lesen von Daten mit der CPU 68008 im schnellsten Betriebsmode nur zwei Taktzyklen benötigt werden. Dadurch erhöht sich gegenüber einer CPU Z80 beim Lesen von Daten der Durchsatz (bei gleichem Systemtakt). Weiterhin wurde bei der Entwicklung der DTACK-Logik berücksichtigt, daß die CPU 68008 auch „Read-Modify-Write"-Zyklen durchführen kann. Diese Zyklen werden bei dem TAS-Befehl („Test And Set") der 68008 durchlaufen.

Einen sehr großen Vorteil gegenüber anderen CPU-Bausteinen weist die CPU Z80 durch ihre interne Refresh-Logik auf. Das Vorhandensein einer solchen Logik vereinfacht in jedem Fall den Einsatz von dynamischen RAMs. Da die CPU 68008 keine derartige Refresh-Logik besitzt, wurde eine solche Logik diskret aufgebaut. Die Refresh-Logik erzeugt jedoch keine Refresh-Adressen, so daß nur dynamische RAM-Karten mit eigenem Refresh-Controller Verwendung finden können.

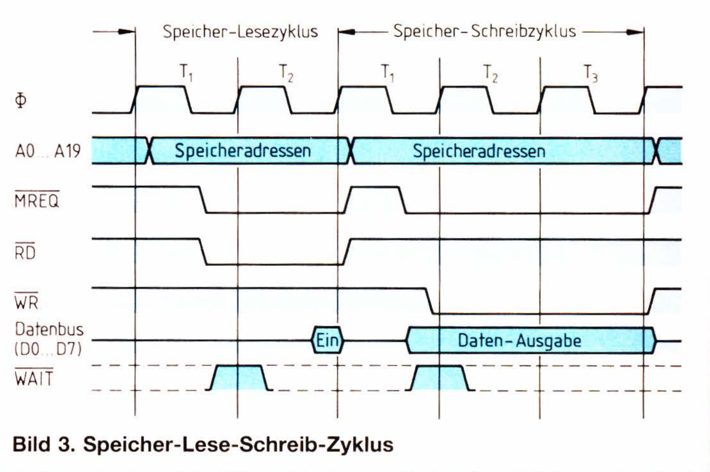


Bild 3. Speicher-Lese-Schreib-Zyklus

Weil die Refresh-Logik nur nach jedem 16. Objectcode-Fetch einen Refreshzyklus durchläuft, werden im Mittel nur 4% der Arbeitszeit der CPU 68008 für Refreshzwecke benötigt.

Da der gesamte adressierbare Speicherbereich der CPU 68008 eine Memory-Mapped-Organisation besitzt, mußte eine Möglichkeit geschaffen werden, die Position der bei einer CPU Z80 verfügbaren 256 E/A-Adressen im Speicherbereich zu definieren. Die E/A-Logik macht es möglich, über Drahtbrücken die 256 E/A-Adressen in Einheiten von 256 Byte über den gesamten adressierbaren Speicherbereich frei einzustellen. Diese 256 E/A-Adressen können nur über Datenzugriffe erreicht werden. Das bedeutet, daß parallel zu den E/A-Adressen auch Objectcodes stehen können.

Auf der 68008-CPU-Karte sind insgesamt 8 KByte EPROM und 8 KByte RAM vorhanden. EPROM und RAM nehmen die ersten 16 KByte des vorhandenen Adreßraums ein. Sie sind hauptsächlich für Bootstrap-Zwecke gedacht, können jedoch, je nach Systemkonfiguration, auch anders genutzt werden.

Das 8 KByte große RAM kann man durch ein EPROM gleicher Speicherkapazität ersetzen, so daß maximal 16 KByte Festwertspeicher auf der CPU-Karte zur Verfügung stehen können.

Da die meisten Betriebssysteme hauptsächlich in RAM-Speichern arbeiten, ist es sinnvoll, den Bootstrap-Bereich ausblenden zu können, so daß der dazu parallel liegende RAM-Bereich genutzt werden kann. Auf der CPU-Karte kann man dieses mit Hilfe eines „Write-EPROM"-Befehls erreichen.

Die CPU 68008 verfügt über die Möglichkeit, prioritätsgestaffelte Vektor-Interrupts auf drei verschiedenen Ebenen zu bearbeiten. Interrupts können beim VAMOS-80-Bus über die drei Leitungen INT1, INT2 und NMI ausgelöst werden. Dem Signal INT1 ist die niedrigste Priorität zugeordnet, während das Signal NMI die höchste Priorität besitzt.

Die Interrupt-Prioritätslogik empfängt die Interrupt-Signale, die die CPU-Karte über den Bus erreichen und leitet den Interrupt mit der höchsten Priorität an die CPU weiter. Am Ende einer Interrupt-Routine steht in einem Z80-Programm ein „RETI"-Befehl. Der „RETI"-Befehl (ED4D Hex) kann mit der CPU 68008 softwaremäßig nachgebildet werden.

Die CPU 68008 verfügt über eine bidirektionale Resetleitung. Je nach Betriebszustand der CPU kann die Resetleitung sowohl als Eingang als auch als Ausgang arbeiten. Befindet sich die Resetleitung im Eingangs-Modus, so kann die CPU über diese Leitung durch einen Systemreset in einen definierten Anfangszustand versetzt werden. Durch den Reset-Befehl der CPU wird die Resetleitung in den Ausgabe-Mode umgeschaltet. Das heißt, die CPU kann über die Reset-Logik einen Reset auslösen, um zum Beispiel Peripherie-Bausteine neu zu initialisieren, ohne selbst neu initialisiert zu werden.

Alle Signalleitungen, die von der CPU-Karte auf den Bus geführt werden, als auch alle Signalleitungen, die vom Bus auf die CPU-Karte geführt werden, sind voll gepuffert. Jede Steuerleitung wird auf der Busseite zusätzlich über einen 4,7-kΩ-Widerstand an +5 V gelegt. Dadurch wird erreicht, daß bei Übernahme des Busses durch eine DMA während der Totzeit keine undefinierten Signale auf dem Bus anliegen. Zusätzlich zu den Steuerleitungen sind auch die oberen vier Adressen A16...A19 zum Bus hin mit 4,7-kΩ-Widerständen an +5 V gelegt. Diese Pull-Up-Widerstände werden benötigt, um die obersten vier Adressen auf einen definierten TTL-Pegel zu legen, für den Fall, daß eine DMA-Karte den Bus übernimmt und selbst keine 20 Adreßleitungen hat. Durch diese Maßnahme verbleibt dem Benutzer die Möglichkeit, die oberste 64-KByte-Seite des Speicherbereichs mit einem DMA zu erreichen, der lediglich 16 Adreßleitungen zur Verfügung stellt.

Da die 68008-CPU-Karte voll DMA-fähig ausgelegt ist, wird eine Logik benötigt, die die Puffer zum richtigen Zeitpunkt aktiviert beziehungsweise desaktiviert. Diese Aufgabe übernehmen Puffersteuerung und Bus-Request-Logik.

Zeitverhalten der 68008-CPU-Karte

Bild 2 zeigt die Signalform der CPU-Karte für Objectcode-Fetch und Refresh-Zyklen. T3 und T4 können entfallen, wenn das System mit statischen RAMs ausgerüstet ist und die Refresh-Logik deaktiviert wurde. Beim Lesen und Schreiben von Speicher-Bausteinen weist das Zeitverhalten der 68008-CPU-Karte gegenüber dem Z80-Timing einige Änderungen auf. Der Lese-Zyklus benötigt nur noch zwei Taktperioden. Die Signale MREQ, RD und WR wurden in ihrem zeitlichen Verlauf geringfügig verändert, um die zur Verfügung stehende Zykluszeit optimaler zu nutzen. *Bild 3* verdeutlicht, in welcher Form die Optimierung vorgenommen wurde und zeigt Schreib- und Lesezugriffe in der schnellsten Betriebsart.

Bild 4 und *Bild 5* zeigen den zeitlichen Verlauf von Lese- und Schreibzugriffen auf Peripherie-Bausteine, wie er auf der 68008-CPU-Karte erzeugt wird. Beiden Diagrammen kann man entnehmen, daß auch bei den E/A-Zugriffen eine Optimierung des Z80-Timings vorge-

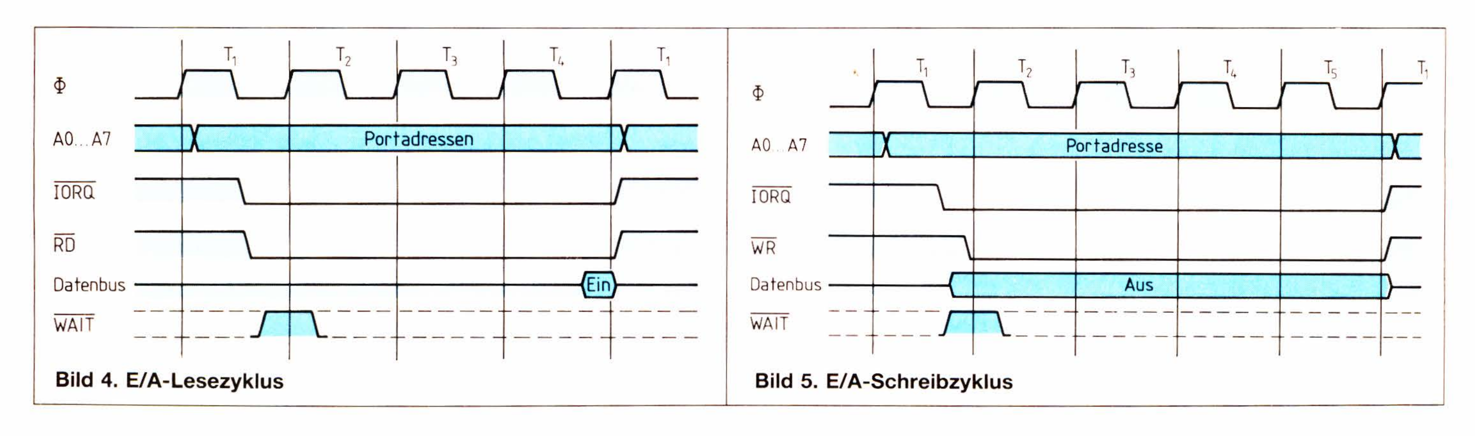


Bild 4. E/A-Lesezyklus

Bild 5. E/A-Schreibzyklus

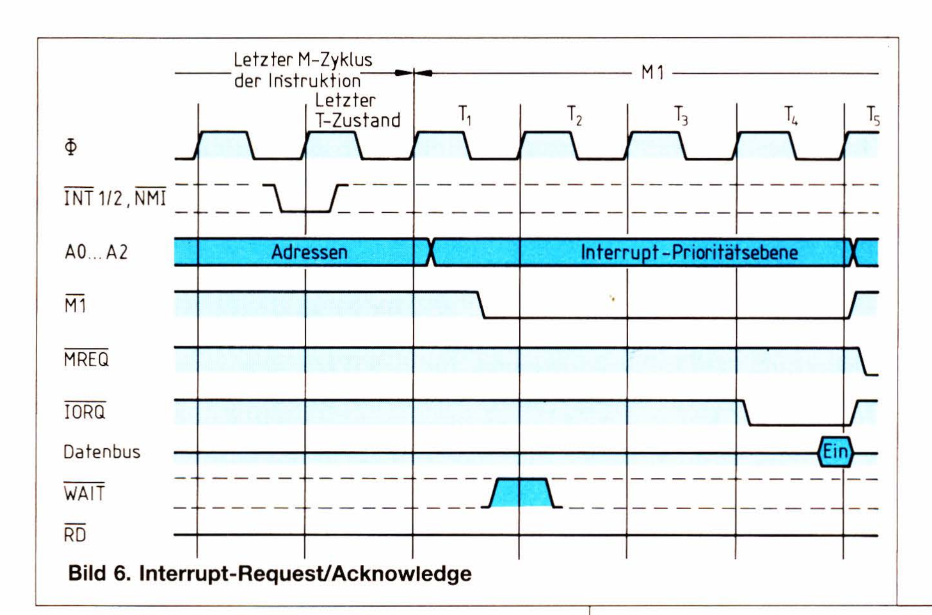


Bild 6. Interrupt-Request/Acknowledge

nommen wurde. *Bild 5* läßt erkennen, daß der E/A-Schreibzyklus um eine Taktperiode länger ist als der E/A-Lesezyklus. Die Verlängerung des E/A-Schreibzyklus wurde im Hinblick auf den Einsatz einer 12,5-MHz- oder 16-MHz-Version der CPU 68008 vorgenommen.

Bild 6 zeigt den zeitlichen Verlauf eines Interrupt-Requests an die 68008-CPU-Karte und deren Acknowledgezyklus. Der Zeitverlauf ist für alle drei Interrupt-Ebenen identisch. Beim Interrupt-Acknowledge-Zyklus unterscheidet sich der zeitliche Ablauf bei der 68008-CPU-Karte ebenfalls geringfügig von dem Z80-Timing. Über die Adreßleitungen A0...A2 wird der Prioritätslevel des zu bearbeitenden Interrupts angezeigt. Die CPU 68008 ist nur für Vektor-Interrupts ausgelegt. Das entspricht dem „IM2-Mode" der CPU Z80.

Bild 7 beschreibt das zeitliche Verhalten eines Bus-Requests und des darauf folgenden Bus-Acknowledge. Das Timing, das von der 68008-CPU-Karte erzeugt wird, ist vollkommen mit dem Timing einer CPU Z80 identisch. Die Signale MREQ, RD, WR, IORQ und RFSH werden während der Totzeit über Pull-Up-Widerstände auf High-Pegel gehalten.

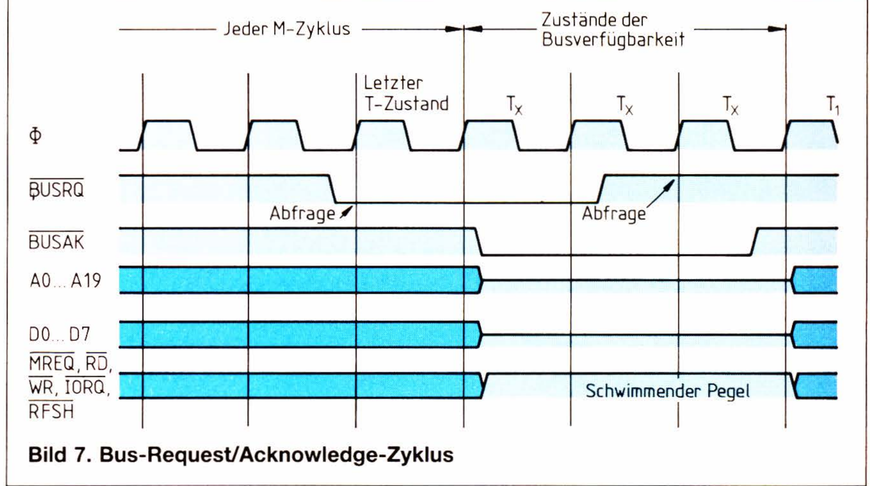
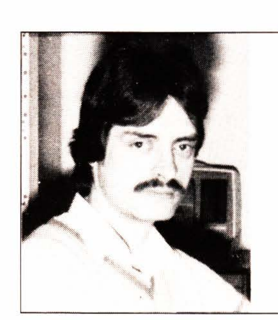


Bild 7. Bus-Request/Acknowledge-Zyklus

Für den Aufbau eines lauffähigen Systems mit der 68008-CPU-Karte benötigt man eine Speicherkarte mit wenigstens 128 KByte RAM, eine Karte mit Floppy-Disk-Controller und eine Karte mit einer oder mehreren Serienschnittstellen. Ein solches System wurde vom Autor aufgebaut und das Betriebssystem CP/M-68K darauf implementiert. Eine Leistungssteigerung gegenüber einem Z80-System ist ganz deutlich spürbar. Unberücksichtigt geblieben ist dabei jedoch noch die Tatsache, daß die CPU 68008 einen linearen Adreßbereich von 1 MByte und einen sehr viel effektiveren Befehlssatz als die CPU Z80 aufweist, so daß, je nach Geschick des Programmierers, insgesamt mit einer Leistungssteigerung von 100...500 % bei gleichem Systemtakt gerechnet werden kann. In einigen Fällen, zum Beispiel bei 32-Bit-Multiplikationen und bei 32-Bit-Divisionen, beträgt die Leistungssteigerung mehr als das 100fache. Verwendet man eine CPU 68008, die mit 12,5 MHz oder mit angekündigten 16 MHz arbeitet, so erreicht man sicherlich die Grenzen eines 8-Bit-Systems. In vielen Anwendungsbereichen wird jedoch ein solches System in bezug auf Verarbeitungsgeschwindigkeit und Speicherbedarf vollkommen ausreichend sein, so daß nicht die Notwendigkeit für ein 16-Bit-System besteht.

Die Entwicklung des Multi-Mikroprozessor-Multi-Bus-Systems im Institut für angewandte Physik der Universität Münster wird von Herrn *Prof. Dr. Rainer Kassing* geleitet. Die 68008-CPU-Karte ist über die Systec GmbH, Münster, als „CPU-122" erhältlich.



Diplom-Physiker Hans-Jürgen Nischik stammt aus der Nähe von Osnabrück. Er studierte an der Universität Münster Physik und promoviert dort zur Zeit mit einem Beitrag zur Entwicklung eines Multi-Prozessor-, Multi-Bus-Systems.
Hobbys: Musik, System-Entwicklungen

Literatur

[1] Motorola Microprocessors Data Manual 1983.
[2] Zilog Z80A-CPU: Technical Manual.
[3] Systec Datenblatt: Vamos-80 Bus.

Dipl.-Phys. Dr. Thomas Denker

UNIX für die M68000-Familie

Nachdem das Betriebssystem UNIX im vergangenen Jahrzehnt bei so erfolgreichen Mini- und Superminicomputern wie PDP-11 oder VAX-11 weite Verbreitung gefunden hat, erkannte man auch seine Eignung als Softwareumgebung für Mikrocomputer. Inzwischen standen nämlich Prozessoren zur Verfügung, deren Leistungsfähigkeit mit der der Zentraleinheiten von Minicomputern vergleichbar ist. Speziell die Prozessorenfamilie M68000 ist zur bevorzugten Hardwareumgebung für solche Systeme geworden: Rund 80 % aller UNIX-Neuentwicklungen basieren auf den Prozessoren MC68000 oder MC68010. Mit dem Erscheinen von SYSTEM V/68 wird nun das „Portieren", also der Transfer des Systems von der ursprünglichen VAX-11- oder PDP-11-Umgebung auf M68000-gestützte Rechner, in standardisierte Bahnen gelenkt.

Unter den vielen Eigenschaften, die zum Erfolg von UNIX beigetragen haben, sticht die Harmonie zwischen den Grundprinzipien des Systems und der „Philosophie" der eigens für UNIX entwickelten Programmiersprache C besonders ins Auge. Eine ähnliche „Paßgenauigkeit" findet sich auch zwischen UNIX und den Grundzügen der Architektur von M68000-Prozessoren sowie zwischen C und dem für diese Sprache maßgeschneiderten Instruktionssatz dieser Familie.

Ein „Mainframe" für jeden Arbeitsplatz

UNIX gliedert sich in zwei Hauptbestandteile. Da ist zum einen der „Kernel" des Systems, ein Programmstück, das all die Funktionen wahrnimmt, die einzelnen Applikationsprogrammen nicht überlassen werden dürfen, um die Integrität des Systems nicht zu gefährden. Demgegenüber steht ein Paket von rund 500 Dienstprogrammen („Utilities"), deren Leistungsfähigkeit fast allen Ansprüchen technisch-wissenschaftlicher als auch kommerzieller Datenverarbeitung gerecht wird.

Neben den unter UNIX bevorzugten Programmiersprachen C, Fortran, Basic und Assembler werden noch spezielle Compiler unterstützt, u. a. sogenannte „Compiler-Compiler", die dem Anwender die rasche Entwicklung eigener, dedizierter Sprachen erlauben.

Die größte Stärke des Systems zeigt sich bei der Durchführung umfangreicher Software-Entwicklungsprojekte, die mit den konventionellen Hilfsmitteln der Mini- und Mikrocomputerwelt nur schwer in den Griff zu bekommen sind. Während sich nämlich fast alle herkömmlichen Systeme im „klassischen Dreiergespann" von Editor, Compiler und Linker erschöpfen, stellt UNIX wesentlich weitergehende Mittel zur Verfügung. Mit symbolischen Debuggern und Hilfsmitteln zur Syntaxüberprüfung von in C geschriebenen Quellprogrammen wird den Erfordernissen schnellen, fehler-

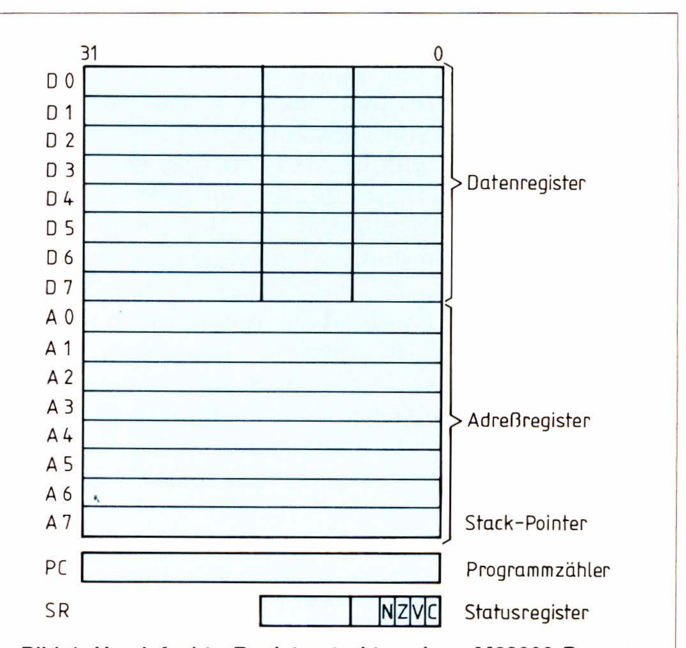


Bild 1. Vereinfachte Registerstruktur eines M68000-Prozessors. Die Register D0...D7 sind vornehmlich zur Aufbewahrung und Verarbeitung von 8, 16 und 32 Bit breiten Datenworten bestimmt, während die Register A0...A7 in der Regel Adressen enthalten. Hinter A7 verbergen sich, je nach Betriebszustand des Prozessors, verschiedene Stackpointer. Als Framepointer dient eines der Register A0...A6

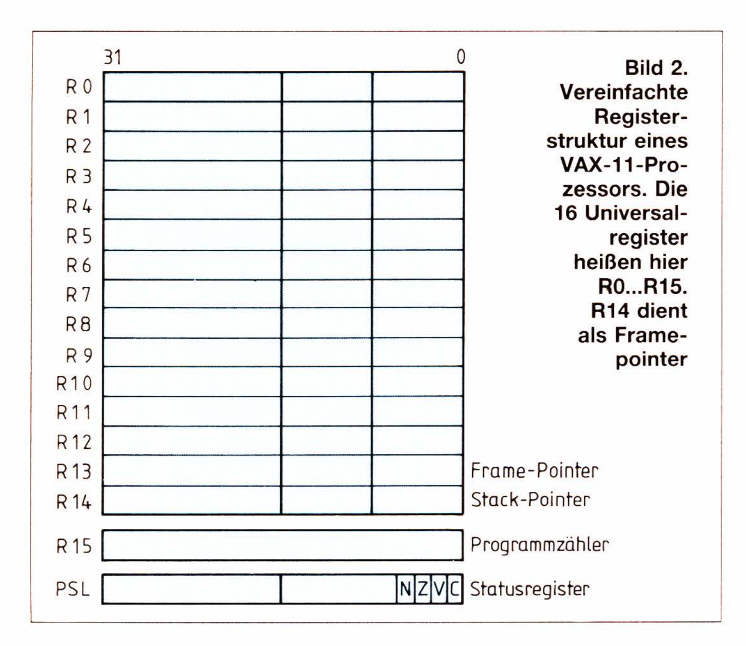
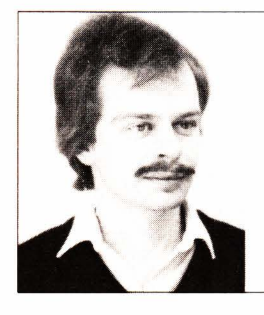


Bild 2. Vereinfachte Registerstruktur eines VAX-11-Prozessors. Die 16 Universalregister heißen hier R0...R15. R14 dient als Framepointer

freien und vor allem kostenkontrollierten Softwareentwurfs Rechnung getragen. In dieselbe Richtung zielt ein „Source Code Control System", das die Unterhaltung sich fortentwickelnder Softwarebibliotheken auch dann noch übersichtlich macht, wenn die Zahl der Files die Grenze von 10^3 zu übersteigen beginnt, wenn also der Punkt erreicht wird, an dem die Software-Entwicklungskosten bislang exponentiell anzusteigen pflegten. Die Durchführung von Software-Entwicklungsprojekten gehobener Komplexität und mit einer größeren Anzahl von Mitarbeitern wird von UNIX stark unterstützt. Das System kommt dabei – u. a. durch integrierte Textprozessoren – dem unumgänglichen Trend nach, der weg vom „Programmieren" hin zum „Softwaredesign" führt.

UNIX ist kein Echtzeit-System. Nichtsdestoweniger, z. T. aber auch gerade deshalb, ist es ein schnelles Betriebssystem. Ein wesentlicher Teil seiner Leistungsfähigkeit kommt dabei aus der Unterhaltung eines Systems von Pufferspeichern, die einen großen Teil der erforderlichen Massenspeicherzugriffe abblocken. Ein logischer Zugriff auf eine Magnetplatte führt nicht notwendig zu einem physikalischen Ansprechen des Peripheriegerätes. Dieses Verfahren wäre zwar bei einem Echtzeit-Betriebssystem in aller Regel nicht akzeptabel, erlaubt es aber UNIX, mit Hilfe eines Teils des verfügbaren Halbleiterspeichers den traditionellen Engpaß Plattenzugriff zu umgehen. Anders gesagt: UNIX honoriert viel Speicher mit hohem Durchsatz – selbst dann, wenn nur ein Benutzer auf das System zurückgreift. Mit preisgünstigen VLSI-RAMs und den großzügig dimensionierten linearen Adreßräumen ist die Implementierung großer Arbeitsspeicher ($\geqslant 2^{16}$ Byte) und die damit einhergehende Freisetzung der vollen Leistungsfähigkeit von UNIX einfach geworden.



Dr. phil. nat. Thomas Denker ist in Wiesbaden geboren und studierte in Frankfurt Physik. Von 1978 ab beschäftigte er sich als wissenschaftlicher Mitarbeiter der Universität Frankfurt mit Fragen der digitalen Signal- und Sprachverarbeitung sowie der Anwendung von Array-Prozessoren. Seit 1983 ist er für die Unterstützung von UNIX bei Motorola Microsystems in München zuständig.

Die Programmiersprache C

Das Betriebssystem UNIX ist fast vollständig in der höheren Programmiersprache C geschrieben. Obwohl diese Sprache den Einsatz von Assembler-Code weitestgehend überflüssig macht, kann C-Quellcode als wirklich „portabel" gelten. Die Übertragung des in C geschriebenen Betriebssystems in eine andere Hardwareumgebung wird dadurch relativ leichtgemacht.

C ist eine Programmiersprache, bei deren Definition Bestandteile anderer höherer Sprachen eingeflossen sind. C ist blockstrukturiert wie Pascal, enthält aber auch Elemente von Fortran oder PL1. Die sinnvolle Aufteilung des benutzten Datenspeichers in lokale und globale Bereiche sowie die Bereitstellung von statischen und dynamischen (stackresidenten) Variablen paßt sich in natürlicher Weise den Grundstrukturen pointer- und stackorientierter Maschinen wie einer PDP-11 oder VAX-11 an. Da alle Prozessoren der Familie M68000 genau wie eine VAS-11 auf einem Block von 16 Universalregistern mit jeweils 32 Bit Universalregistern aufbauen, sind bei der Konstruktion von C-Compilern für M68000-Prozessoren keine „Verrenkungen" erforderlich. Die *Bilder 1* und *2* zeigen vereinfachte Schemata des Registeraufbaus des M68000- und der VAX-11.

Ähnliche Parallelen wie beim Registeraufbau finden sich bei den intrinsischen Datenstrukturen eines VAX-11- bzw. eines M68000-Prozessors. Bei beiden Maschinen wird byteweise adressiert. Die Daten werden bevorzugt als ein-, zwei- oder vier-Byte-Worte vom oder zum Speicher transferiert. Diesen Vorgaben der Hardware kommt C dadurch nach, daß als primäre Datentypen „char" (immer = 1 Byte), „short" (meist = 2 Byte) und „int" = 4 Byte) angeboten werden. Schließlich erlauben die 32 Bit breiten Daten- und Adreßregister den zwanglosen Umgang mit Daten und Zeigern auf solche Daten. C nutzt diese Möglichkeit intensiv und kann sich dabei auf solche Adressierungsarten der M68000-Prozessoren stützen, die die bequeme Unterhaltung der am häufigsten anzutreffenden Datenstrukturen wie Arrays, Stacks und Queues erlauben. Die *Bilder 3* und *4* zeigen ein kleines Stück C-Quellcode und ein äquivalentes Stück M68000-Assembler-Code. Das Beispiel ist so einfach gestaltet, daß zum Verständnis weder Kenntnisse von C noch vom Instruktionssatz eines MC68000 benötigt werden. Die Frage, welcher der zur Verfügung stehenden C-Compiler tatsächlich zu einer solch optimalen Codierung kommt, soll dabei dahingestellt bleiben; das Bei-

spiel soll lediglich die Ähnlichkeit zwischen der „C-Welt" und der Philosophie des M68000-Instruktionssatzes aufzeigen.

Beide Programme gehen am Anfang von zwei Pointern aus, die jeweils auf den Beginn eines Strings (Feld von ASCII-Zeichen) zeigen. Die Aufgabe der Programme besteht nun darin, die beiden Strings zu vergleichen, wobei jedoch höchstens 80 Zeichen zu berücksichtigen sind. Das Aufsetzen der Pointer ist in beiden Beispielen nicht dargestellt. Für den zeitkritischen Schleifenteil kann man offenbar in C mit einer Quellcodezeile und beim M68000 mit nur zwei Assembler-Instruktionen auskommen.

```
count = 80 - 1;                                       /* load counter */
while ( *a1++ == *a2++ && count-- >= 0 ) ;            /* repeat as long as
                                                         characters are equal
                                                         and counter is
                                                         not negative */
```

Bild 3. C-Quellcode zum Vergleich zweier Character-Strings der Länge 80. Nach Ablauf der „While"-Schleife zeigen a1 und a2 hinter die beiden ersten voneinander verschiedenen ASCII-Zeichen oder hinter die achtzigsten Zeichen der beiden Strings

```
       MOVE    80-1,D1          LOAD COUNTER

LOOP   CMP.B   (A1)+,(A2)+      COMPARE BYTES WITH POSTINCREMENT
       DBNE    D1,LOOP          DECREMENT COUNTER AND REPEAT LOOP
*                               UNTIL NEGATIVE OR CHARACTERS NOT
*                               EQUAL
```

Bild 4. M68000-Assembler-Code zum Vergleich zweier Strings der Länge 80 Zunächst wird das Register D1 mit der Zahl 79 geladen. Die zweite Instruktion beim Label „LOOP" vergleicht zwei Zeichen und erhöht gleichzeitig die beiden Zeiger in den Registern A1 und A2. Der nachfolgende „*D*ecrement and *B*ranch if *EQ*ual"-Befehl wiederholt die Prozedur durch einen Sprung nach „LOOP" so lange, bis entweder der Zähler in D1 abgelaufen oder zwei verschiedene Zeichen gefunden sind. Ein 8-Bit-Prozessor MC68008 benötigt bei einer Taktfrequenz von 8 MHz für jeden Schleifendurchlauf 4,25 µs, ein MC68000 mit seinem 16-Bit-Datenbus bei der gleichen Frequenz 2,75 µs. Der virtuelle Prozessor MC68010 benötigt bei einer Taktrate von 10 MHz für jeden Durchlauf noch 1,4 µs. Der ab Mitte des Jahres zur Verfügung stehende Prozessor MC68020 verkürzt diese Zeit sogar noch weiter auf weniger als 1,0 µs

SYSTEM V/68 – ein UNIX-Standard

Der entscheidende Schritt beim Übergang von den Vorläufern (etwa SYSTEM III oder der Version 7) zu UNIX SYSTEM V besteht nun nicht in den zahlreichen funktionellen Verbesserungen, sondern vielmehr in der Tatsache, daß mit SYSTEM V erstmals ein von AT&T kommerziell verwertetes und entsprechend unterstütztes Produkt zur Verfügung steht. Galt nämlich UNIX bis vor nicht allzulanger Zeit als ein „Instrument von Wissenschaftlern für Wissenschaftler", so ist spätestens mit SYSTEM V klar geworden, daß AT&T mit UNIX ein großes Stück des Softwaremarkts für kleine und mittlere Rechner für sich beanspruchen wird. Die Kooperation mit einem großen amerikanischen Halbleiterhersteller ist dabei als Teil einer Strategie zu sehen, die Hardwarebasis von UNIX so weit wie möglich zu spannen. Die obenerwähnte Tatsache, daß das System beinahe vollständig in der portablen Programmiersprache C codiert ist, wirkt sich dabei außerordentlich günstig aus. Mit SYSTEM V/68 wird nun die erste solche Gemeinschaftsproduktion von AT&T und einem Halbleiterhersteller vorgestellt. Dem SYSTEM V/68 von Motorola werden SYSTEM-V-Standards von anderen Herstellern folgen.

Es ist absehbar, daß die Anwender der M68000-Serie von 16- und 32-Bit-Mikroprozessoren, die bislang UNIX in voneinander unabhängigen und nicht standardisierten Ports auf ihre jeweiligen Hardwareumgebungen gebracht, mehr und mehr von den „wilden", UNIX- und UNIX-artigen Systemen zu SYSTEM V/68 übergehen.

Nach einer Überprüfungsphase durch AT&T dürfte dann der sog. „Generic Port" für den M68000-Prozessor zur Grundlage aller weiteren UNIX-SYSTEM-V-Entwicklungen für M68000-Systeme werden. Für die Anwender resultiert daraus Sicherheit für die Zukunft ihrer auf M68000-Technik und UNIX basierenden Produkte: weder Motorola noch AT&T werden bei der Fortentwicklung ihrer Prozessoren bzw. beim Ausbau von SYSTEM V Schritte unternehmen, die eine weitere enge Verbindung der Produkte behindern könnten.

Kritischer Punkt: Speicherverwaltung

SYSTEM V/68 ist auf das Multi-User-Entwicklungssystem EXORmacs zugeschnitten. Dieses System baut auf einem MC68000-Prozessor mit einer diskreten Speicherverwaltungseinheit (MMU) auf. Eine zweite Version steht auf dem Entwicklungssystem VME/10 zur Verfügung. Die Architektur dieses Tischcomputers beruht auf der Kombination des virtuellen Mikroprozessors MC68010 mit einem oder mehreren MMU-Bausteinen MC68451. Die vollständige Entkopplung von Prozessor und MMU erlaubt dabei eine freie Entscheidung zwischen einer segmentierten, einer Demand-Page- oder einer Demand-Segment-Speicherverwaltung. Alle mit dem jeweils angewandten MMU-Mechanismus in Beziehung stehenden Teile des Kernels sind in einem MMU-Treiber zusammengefaßt, so daß ein Wechsel des Speicherverwaltungsalgorithmus sich nicht komplizierter gestaltet als etwa der Übergang zu einem anderen Massenspeichersystem. Mit anderen Worten: Der Arbeitsspeicher der jeweiligen Maschine ist zu einer Ressource des Computers geworden.

Literatur

[1] *Harteloo, F.*: Development Systems and VAX-like chips suit UNIX. Electronics, Sept. 8, 1983.

[2] *Pol, B.*: Die Sprache C. ELEKTRONIK 1983, H. 2, S. 43...49, H. 3, S. 53...59, H. 4, S. 61...65.

Komplexe Multiplikation in kurzer Zeit

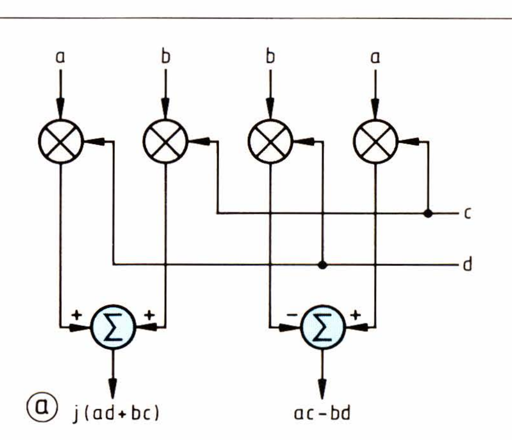
Bei vielen Grafik- und Signalverarbeitungs-Algorithmen sind komplexe Multiplikations-Operationen erforderlich. Wenn diese Aufgabe mit einer flexiblen MPU wie dem Typ 68000 gelöst wird, kann die erforderliche Zeit relativ lang werden, denn für eine komplexe Multiplikation muß man vier Einzelmultiplikationen ausführen. Jede Multiplikation benötigt bis zu 70 Taktzyklen. Merklich verkürzen läßt sich die Multiplikationszeit mit Hilfe eines einfachen Algorithmus. Die komplexe Multiplikation kann so neu strukturiert werden, daß lediglich drei Einzelmultiplikationen durchzuführen sind (*Bild*). Als Ergebnis zeigt sich eine Zeitersparnis von 18 % im Vergleich zur konventionellen Methode. Der Algorithmus lautet:

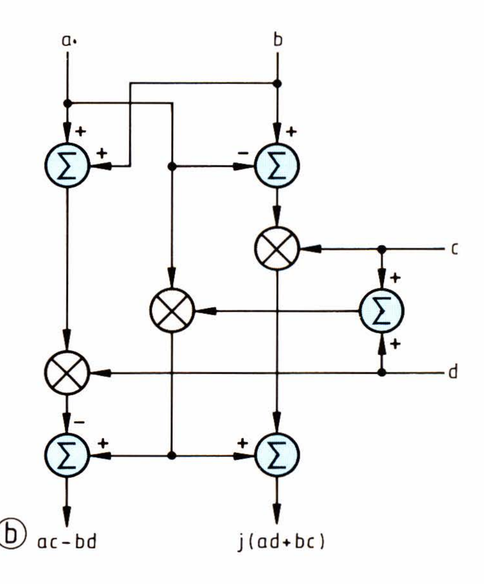
$$(a + jb) \cdot (c + jd) = [a(c + d) - (a + b)d] + j[c(b - a) + (c + d) a] = (ac - bd) + j (ad + bc)$$

Randy Wilhelm

Tabelle der abzuarbeitenden Befehle

konventionell			modifizierte Multiplikation		
MOVEM.W	(A1)+, D1...D4	28	MOVEM.W	(A1)+, D1...D4	(28)
A1 zeigt auf den Speicherplatz von a. b, c und d befinden sich in den darauffolgenden Plätzen					
MOVE .W	D1, D5	(4)	MOVE .W	D1, D5	(4)
MOVE .W	D2, D6	(4)	MOVE .W	D3, D6	(4)
MULS	D4, D1	(70)	ADD .W	D2, D1	(4)
MULS	D4, D2	(70)	SUB .W	D5, D2	(4)
MULS	D3, D5	(70)	ADD .W	D4, D3	(4)
MULS	D3, D6	(70)	MULS	D4, D1	(70)
ADD .L	D1, D6	(4)	MULS	D5, D3	(70)
SUB .L	D2, D5	(4)	MULS	D6, D2	(70)
			ADD .L	D3, D2	(4)
			SUB.L	D1, D3	(4)
Anzahl der Taktzyklen: 324			Anzahl der Taktzyklen: 266		





Mit einer mathematischen Umstellung ergibt sich ein Multiplikationsalgorithmus für komplexe Zahlen, der mit nur drei Einzelmultiplikationen auskommt; a) zeigt die konventionelle Methode, b) zeigt den minimierten Aufbau

Dipl.-Ing. Pal Poth

Verkürzter WAIT-Zyklus beschleunigt 68000-Systeme

Daten-, Adreß- und Steuersignale des Mikroprozessors 68000 sind mit dem Prozessortakt starr gekoppelt. Sind die Peripheriebausteine nicht schnell genug, dann fügt die CPU einige WAIT-Zyklen in den normalen Ablauf ein. Dadurch sinkt die Leistung des Systems. Dieser Beitrag zeigt, daß man die Leistungseinbuße mit einer kleinen Zusatzschaltung in Grenzen halten kann.

Nachdem die Peripherie Adresse und Datum erkannt hat, bestätigt sie das durch die abfallende Flanke des Signals $\overline{\text{DTACK}}$ (*Bild 1*). Geschieht das nicht mindestens um die sogenannte Set-up-Zeit (t_{ASI} = 20 ns bei 8 MHz) vor der abfallenden Flanke von S4, dann fügt der Prozessor WAIT-Zyklen ein. Welche Auswirkungen das auf die maximale Betriebsfrequenz hat, zeigt *Tabelle 1*.

Ein normaler Lesezyklus ist beim 68000 vier Taktperioden lang. Der Schreibzyklus dauert fünf Taktperi-

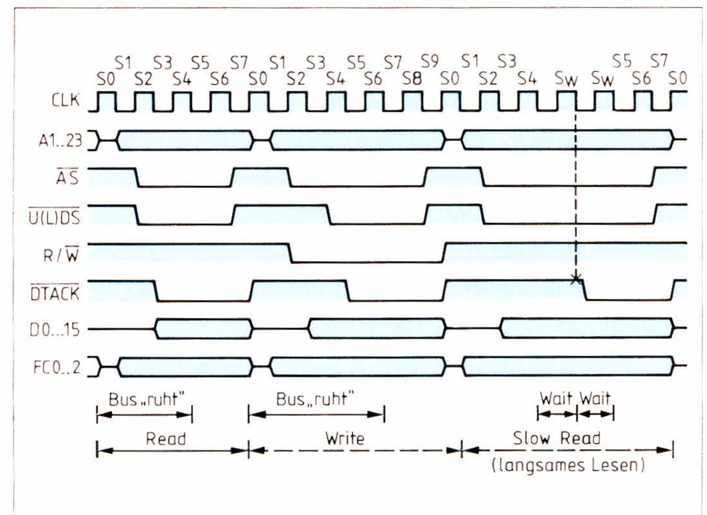


Bild 1. Zeitdiagramm für das Lesen und Schreiben: Bei langsamen Peripheriebausteinen werden WAIT-Zyklen eingefügt

Tabelle 1. Maximale Betriebsfrequenz bei unterschiedlichen RAM- und Treiberbausteinen

RAM-Zugriffszeit t/ns	50	100	150	200	200	250	250	300	350	400	450
Treiber – Schottky	28	28	28	28		28					
Treiber – LS					56		56	56	56	56	56
$t_{AS/RAS}$	7	7	7	7	22	7	22	22	22	22	22
Verzögerung MC68000 (t_{ASI} & Verz. AS)											
10 MHz: 10 + 50 ns	60	60	60								
8 MHz: 15 + 55 ns				70	70	70					
6 MHz: 25 + 65 ns							90	90			
4 MHz: 30 + 75 ns									105	105	105
BUS „ruht" (ns)	145	195	245	305	348	355	418	468	533	583	633
max. Frequenz/MHz (ohne WAIT)	17	12,8	10,2	8,19	7,18	7,04	5,98	5,34	4,69	4,2	3,94
Betriebsfrequenz/MHz (ohne WAIT)	16	12	10	8	7	7	6	5	4	4	3,58

Tabelle 2. Leistungsparameter bei unterschiedlichen RAM- und Treiberbausteinen, bezogen auf die Befehlssequenz „Schreiben-Lesen“

Lösungsweg	Takt-Zyklen	Zeitbedarf	Taktfrequenz	Leistung	Konfiguration
Befehlssequenz (ideal)	17	2125 ns	8 MHz	100 %	200-ns-RAM + Schottky
WAIT bei READ (real)	20	2500 ns	8 MHz	85 %	200-ns-RAM + LS
Reduzierte Taktfrequenz	17	2429 ns	7 MHz	87,5 %	250-ns-RAM + Schottky
Mit Taktdehnung gemäß Bild 3	18,5	2313 ns	8 MHz	92 %	200-ns-RAM + LS

oden. So benötigt die einfachste Befehlssequenz „Lesen-Schreiben“ (entspricht einem Datentransport) einschließlich dem Holen der Befehle insgesamt vier Buszugriffe. Darin ist nur ein einziger Schreibvorgang enthalten. Diese einfache Sequenz dauert somit sage und schreibe 17 Taktperioden, was bei 8 MHz einer Zeit von 2,125 µs entspricht.

In die Überlegungen muß auch die Buskopplung des Systems einbezogen werden, wie sie in *Bild 2* dargestellt ist. Die Schlußfolgerungen aus Tabelle 1 sind in *Tabelle 2* aufgeführt. Ihr liegt die einfache Befehlssequenz „Lesen-Schreiben“ zugrunde. Es fällt auf, daß die Kombination LS-Treiber mit 200-ns-RAMs eine geringere Leistung zur Folge hat als die Lösung mit Schottky-Treibern und herabgesetzter Taktfrequenz. Einen erstaunlichen Leistungswert ergibt die letzte Zeile, die sich auf die Zusatzschaltung in *Bild 3* bezieht.

Wie aus *Bild 4* zu ersehen ist, wird sozusagen auf Wunsch des langsameren Peripheriebausteins der Prozessortakt S4 gedehnt, ohne daß es der Prozessor überhaupt merkt. Dadurch entfallen zusätzliche WAIT-Zyklen, und die Leistungseinbuße bleibt gering.

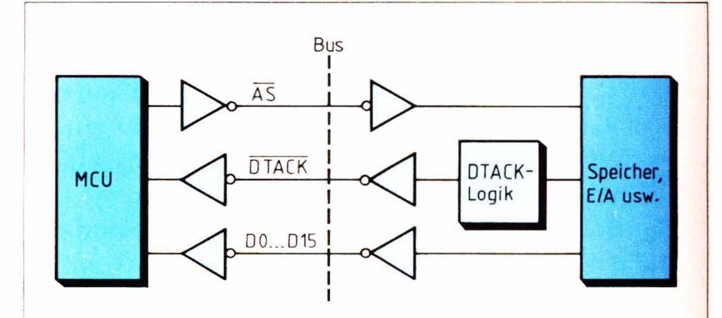


Bild 2. Kopplung des Prozessors mit der Peripherie, vereinfacht dargestellt

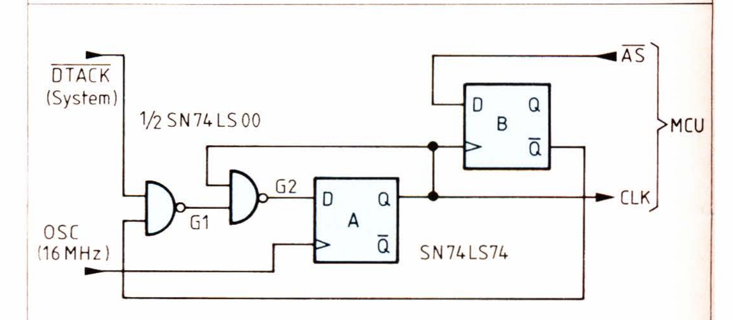


Bild 3. Mit dieser Zusatzschaltung wird der Takt für den Prozessor bei Bedarf gedehnt (S4)

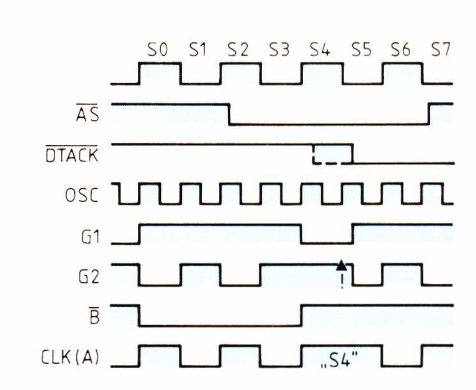


Bild 4. Zeitdiagramm zu Bild 3: Der gedehnte Takt verhindert WAIT-Zyklen und hält so die Leistungseinbuße bei langsamen Peripheriebausteinen in Grenzen



Dipl.-Ing. (TU) Pal Poth, ist geboren in Budapest und studierte Nachrichtentechnik. Danach galt sein Interesse längere Zeit der Kommunikations- und Funknavigationstechnik. Erfahrungen sammelte er auch als technischer Redakteur und Verfasser. Zur Zeit versucht er neben GHz's auch GBytes (per 16-Bit-Fastsysteme) zu „richten“.
Hobbys: Dichten, Orgeln, Kanu, Langlauf, Amateurfunk (KW u. UKW).

Elektronik verstehen, Computer verstehen – mit der ELO fängt der Spaß an moderner Technik erst so richtig an.

Weil Sie in der ELO leicht zu realisierende **Bauanleitungen** finden, die Ihnen Elektronik praktisch nahe bringen und oft **neue Einsatzbereiche für den Computer** erschließen: Messen, Steuern, Regeln, Erfassen, Auswerten.

Und weil ELO Ihnen helfen wird, Hardware in Ihrem Sinne zu verändern – auch als Anfänger.

Daneben finden Sie leicht verständliche **Beiträge zu Grundlagen der Elektronik,** die sehr schnell dazu beitragen, technische Zusammenhänge zu erkennen und zu nutzen.

Im Magazinteil der ELO und in ELO-Reports wird berichtet, **was mit Elektronik und Computern alles möglich ist.** In der Forschung, in der Industrie, im Verkehrswesen, im Umweltschutz...



Die Rubrik ELO-Service bringt regelmäßig **Marktübersichten,** die Ihnen wichtige Orientierungshilfe im immer unüberschaubarer werdenden Markt elektronischer Produkte sein werden.

Und natürlich finden Sie **ELO-Testberichte:** Über Schach- und Home computer, über Video und HiFi-Geräte, über Meßgeräte und alles, was damit im Zusammenhang steht.

Die ELO verbindet die Elektronik mit dem Computer. Wer endlich richtig einsteigen möchte und den Spaß an moderner Technik neu entdecken will, für den gibt's die ELO für DM 4,80 an allen größeren Zeitschriften-Verkaufsstellen.

Noch besser: Sie machen von unserem vorteilhaften Kennenlern-Angebot Gebrauch.

Eine Kennenlern-Karte finden Sie vor dieser Seite.

